Bacheloroppgave

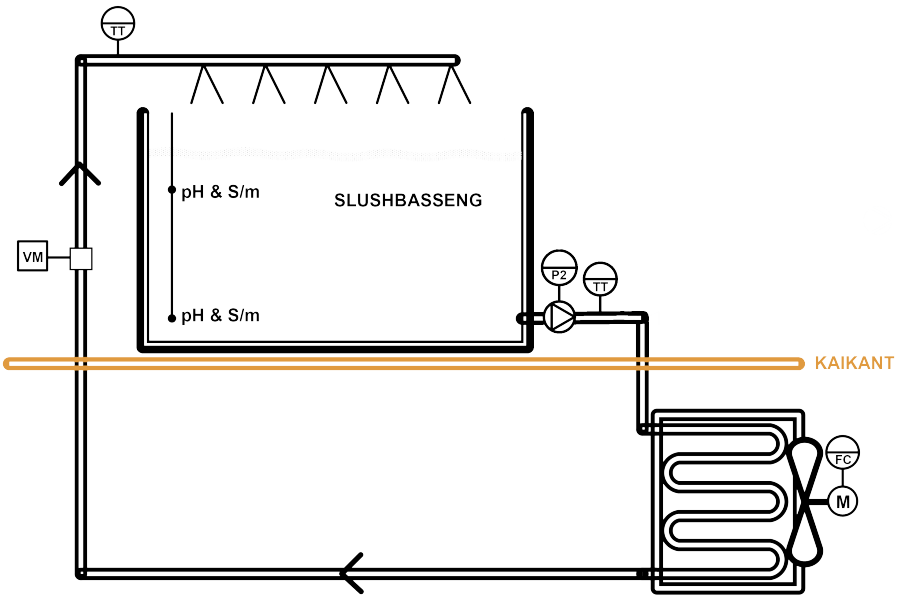
Daniel René Astor, Monika Viken Tornes, Marie Tveter, André Joseph Virani

SMELT

Klima- og miljøvennlig smelting av snø

Bacheloroppgave i Elektroingeniør - BIELEKTRO
Veileder: Kåre Bjørvik
Mai 2024

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for teknisk kybernetikk



VisionTech AS

NTNU
Kunnskap for en bedre verden

VisionTech

Daniel René Astor, Monika Viken Tornes, Marie Tveter, André Joseph Virani

SMELT

Klima- og miljøvennlig smelting av snø

Bacheloroppgave i Elektroingeniør - BIELEKTRO
Veileder: Kåre Bjørvik
Mai 2024

Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for teknisk kybernetikk

NTNU
Kunnskap for en bedre verden

NTNU

Oppgavetittel (norsk og engelsk):

SMELT – Klima- og miljøvennlig smelting av snø

Forfattere:

Daniel René Astor, Monika Viken Tornes, Marie Tveter, André Virani

Prosjektnummer:	E2401
Innleveringsdato:	20.mai
Gradering:	[ ] åpen [x] lukket

Studium:	Elektroingeniør
Studieretning:	Automatisering og robotikk, Elkraft og bærekraftig energi
Veileder internt:	Kåre Bjørvik
Institutt:	Institutt for teknisk kybernetikk
Oppdragsgiver:	VisionTech AS
Kontaktperson:	Runar Holte

Sammendrag (norsk og engelsk):

Norsk: Bacheloroppgaven er del av et større prosjekt for å finne en miljøvennlig måte å smelte brøytesnø på. Dagens metoder skader klimaet og miljøet, og Oslo og Trondheim ønsker en bedre løsning gjennom SMELT-konkurransen. Rebel Garden AS og AF Gruppen ASA, med VisionTech AS, deltar i prosjektet. Deres idé er å bruke sjøvann til å smelte snø ved hjelp av en varmeveksler. En slush av brøytesnø og vann pumpes gjennom veksleren, der varme overføres fra sjøvannet til slushen. Den tekniske løsningen skal gjennom en testperiode, og innsamlet data skal evaluere effektiviteten og videreutvikle prosjektet.

English: The bachelor's thesis is part of a larger project to find an environmentally friendly way to melt snow. Today's methods are harmful to the climate and the environment, and Oslo and Trondheim wants a better solution through the SMELT competition. Rebel Garden AS and AF Gruppen ASA, with VisionTech AS, are participating in the project. Their idea is to use seawater to melt snow using a heat exchanger. A slush of snow and water is pumped through the exchanger, where heat is transferred from the seawater to the slush. The technical solution will go through a test period, and the data collected will evaluate its effectiveness and further develop the project.

Stikkord norsk:

- Klima & Miljø
- Brøytesnø
- Automasjon
- PLS
- HMI
- Datalogging & visualisering

Stikkord engelsk:

- Climate & Environment
- Snow ploughing
- Automation
- PLC
- HMI
- Data logging & visualization

Forord

Denne rapporten er sluttproduktet av en bacheloroppgave, utarbeidet av fire studenter ved NTNU fra studiet "Elektroingeniør" med studieretningene "Automatisering og robotikk" og "Elkraft og bærekraftig energi". Bacheloroppgaven tilsvarer 20 studiepoeng, og er utarbeidet vårsemesteret 2024. Leseren anbefales å ha grunnleggende teknisk forståelse innen reguleringsteknikk, PLS og datakommunikasjon.

Oppgaven er utarbeidet i samarbeid med VisionTech AS, og innebærer å utvikle et automasjonsanlegg med mulighet for datalogging og visualisering. Prosjektet er også en del av en større konkurranse, noe gruppen er veldig takknemlig for å ha fått anledning til å delta i, og er spent på å følge med på videre.

Prosjektet har vært lærerikt, og gruppen har utviklet bred kompetanse, samt fått muligheten til å ta i bruk teori og erfaring fra flere emner i studietløpet. Tverrfagligheten i gruppen har vært nyttig og alle har hatt styrker å bidra med.

Gruppen ønsker å rette en takk til Lasse Iversen ved VisionTech AS for opplæring og oppfølging gjennom perioden med skapbygging, instrumentering og hjelp under testperioden. Gruppen vil også takke Vidar Kvistad ved Betonbygg Trøndelag AS for godt samarbeid som kranoperatør under testperioden. I tillegg vil gruppen takke Hege Berdahl ved VisionTech AS for god rådgivning rundt bachelorskriving og prosjektorganisering.

En stor takk til vår veileder Kåre Bjørvik ved NTNU, og våre oppdragsgivere ved VisionTech AS og Rebel Garden AS; Runar Holte og Lars Skrøvseth, som alle har vært til stor hjelp gjennom prosjektperioden. Det har vært en fryd å utforme prosjektoppgaven i samarbeid med dere. Gruppen er også svært takknemlig for gjestfriheten til VisionTech som har stilt med kontor i egne lokaler.

Til slutt vil vi takke vår familie, nære og kjære for støtten underveis.

Sammendrag

Bacheloroppgaven tar del i et større prosjekt der målet er å finne en klima- og miljøvennlig måte å smelte brøytesnø på. Dagens problem er at måten brøytesnø fra større byer håndteres på, har en skadelig effekt på klima og miljø. Dagens løsninger fører blant annet til at miljøgifter renner ut i naturen og har negativ påvirkning på økosystemer. Trondheim og Oslo kommune ønsker dermed å finne en bedre løsning, gjennom konkurransen SMELT.

Rebel Garden AS og AF Gruppen ASA er en av to konsortium i SMELT-konkurransen, med bistand fra VisionTech AS. Første fase av prosjektet ble gjennomført i 2022. Denne bacheloroppgaven tar del i Fase 2 av prosjektet.

Fase 2 av SMELT-prosjektet tar i bruk sjøvann for å smelte snø. Snøsmelteanlegget består av et basseng og en varmeveksler som ligger senket ned i sjøvann. En blanding av brøytesnø og vann (slush) pumpes gjennom veksleren, slik at varme transporteres fra sjøvannet (5℃) til slushen.

Den tekniske løsningen er et automasjonsanlegg, med tilhørende instrumentering, samt et IT-system for logging og visualisering av data. Den tekniske løsningen skal tas i bruk under en test-periode på 9 dager, der prosjektgruppen skal teste sin tekniske løsning, samt gjøre målinger av smelteprosessen.

Den samlede dataen blir senere tatt i bruk for å avgjøre hvor godt løsningen fungerte. Dette danner grunnlaget for hvordan løsingen kan videreutvikles i kommende faser av SMELT-prosjektet.

Abstract

The bachelor's thesis is part of a larger project where the goal is to find a climate and environmentally friendly way to melt snow. The current problem is that the way snow from major cities is handled has a detrimental effect on the climate and the environment. Among other things, current solutions lead to pollutants running into nature and have a negative impact on ecosystems. Trondheim and Oslo municipalities want to find a better solution through the SMELT competition.

Rebel Garden AS and AF Gruppen ASA are one of two consortia in the SMELT competition, with assistance from VisionTech AS. The first phase of the project was completed in 2022. This bachelor's thesis takes part in Phase 2 of the project.

Phase 2 of the SMELT project utilises seawater to melt snow. The snow melting plant consists of a pool and a heat exchanger submerged in seawater. A mixture of snow and water (slush) is pumped through the exchanger so that heat is transported from the seawater (5℃) to the slush.

The technical solution is an automation system with associated instrumentation, as well as an IT system for logging and visualizing data. The technical solution will be used during a test period of 9 days, where the project team will test their technical solution, as well as make measurements of the melting process.

The collected data will later be used to determine how well the solution worked. This forms the basis for how the solution can be further developed in future phases of the SMELT project.

Terminologi

AI/DI Analog Inngang / Digital Inngang

AO/DO Analog Utgang / Digital Utgang

API Application Programming Interface - Programmeringsgernsesnitt som tillater ulike programvarer å kommunisere og dele data

ARIMA Autoregressive Integrated Moving Average

AWS Amazon Web Services

BAT Best Avaliable Technique

CPU Central Processing Unit (Hovedprosessor)

CSV Comma-separated values - Tillater data å bli lagret i tabellformat.

DB Datablokk i Siemens TIA Portal

DIN-skinne Metallskinne som blir brukt for montering av effektbrytere og industrielt kontrollutstyr i utstyrsstativ

DP-Master Styringsenhet i et Profibus-DP nettverk

FB Funksjonsblokk i Siemens TIA Portal

FC Funksjon i Siemens TIA Portal

FOPTF Første Ordens Prosess med Tidsforsinkelse

HMI Human-Machine Interface - Menneske-maskin interaksjon, brukergrensesnitt

HiveMQ Platform for MQTT

ISO/OSI Open Systems Interconnection, definert av Den Internasjonale Standardiseringsorganisasjonen

IT Informasjonsteknologi

JSON JavaScript Object Notation - Tekstbasert kodespråk for datautveksling

KDE Kernel Density Estimation - Kjernetetthetsestimering

LMQTT MQTT-klient bibliotek i TIA-portal

LTI Linear Time Invariant

MAC-adresse Media Access Control address

MATLAB Matematikk og simuleringsprogram

MIMO Multiple Input Multiple Output - Multivariabelt system

MQTT Message Queuing Telemetry Transport

MQTT Emne MQTT Topic

MQTT Megler MQTT broker

Nettgrensesnitt Måte å samhandle med en enhet, et program eller en tjeneste gjennom en nettleser eller en webapplikasjon

NORVAR "Norsk vann"

NGI Norges Geotekniske Institutt

Node-RED Høynivå flyt-basert programmeringsverktøy for å koble hardware med databaser og annet software

OB Organisasjonsblokker

OT Operasjonell teknologi

Pandas Dataramme Datastrukturer fra Pythonbiblioteket Pandas

PLS/PLC Programmerbar Logisk Styring

PROFINET Process Field Network

Profibus-DP Process Field Bus - Decentralised Peripherals

Python Høynivå programmeringsspråk

QoS Quality of Service

RTD Resistance Temperature Device

SCADA Supervisory Control And Data Acquisition - System for styring og overvåking

SISO Single Input Single Output

SQL Structured query language

Standard Telegram 1 Sett med predefinerte datapakker for kommunikasjon over PROFINET

TIA Totally integrated automation

Tynnfilmmotstand Oversatt: Thin film resistor

Vaierviklet motstand Oversatt: Wire wound resistor

Innholdsfortegnelse

1 Innledning

Dette kapittelet presenterer bakgrunnen for prosjektet, oppgaveteksten gitt av VisionTech AS, samt en problemstilling med tilspisset målsetting. Relevante bærekraftsmål blir også lagt frem.

1.1 Bakgrunn

Brøytesnø fra byene er ofte forurenset med miljøgifter, partikler, mikroplast, sand, grus, salt og søppel [1]. I dag blir denne snøen etterlatt som snøhauger i bygatene eller kjørt ut til deponier utenfor byene. I enkelte tilfeller blir snøen også dumpet direkte ut i sjøen [2], som i følge Miljødirektoratet utgjør en større miljøskade og forurensing enn ved deponering på land [3]. Nylige eksempler på dette er fra januar 2024 da det kom ekstreme mengder snø på Sør-Østlandet, der Larvik og Kragerø kommune bestemte seg for å dumpe snø i sjøen [4]. I februar 2024 er det også overskrifter fra Molde om utfordringene brøytesnøen skaper når den ikke kan kastes rett på sjøen og tar opp plass i sentrum [5]. Snødeponiene kan heller ikke fungere som en endelig løsning, da snøen bringer med seg forsøpling og miljøgifter som kan smelte ut i bekker og grunn, og skade nærliggende økosystemer [6]. I tillegg er det verken økonomisk eller klimavennlig å transportere snøen lange strekninger ut av byen. Det blir stadig mindre areal til å deponere snø utenfor byene, ettersom byene fortetter å vokser. Snøen må dermed transporteres stadig lenger unna sentrum, noe som tar mye tid og gir høyere klimagassutslipp knyttet til transport [1]. Det er derfor veldig viktig å finne en miljø- og klimavennlig måte å håndtere brøytesnø på.

NCC Snowclean har utført et snøsmelteprosjekt i Oslo med snølekteren S/S Terje som stod klar i drift desember 2011 [7]. Anlegget anklages for å ikke være tilstrekkelig klimavennlig, ettersom lekteren får strømforsyning fra dieselaggregat og dermed utgjør en kilde til klimagassutslipp [8]. Bymiljøetaten i Oslo kommune har også påpekt at grenseverdiene satt for flere av de miljømessige parametrene til smeltevannet har nivåer som gir "dårlig tilstand" eller "svært dårlig tilstand" for kystvann. Til dette svarer NCC blant annet at det har "[...] vært gjort kontinuerlige forbedringer av renseprosessen for å redusere påvirkning på resipient". De sier også at deres anlegg "[...] representerte 'best available technique' (BAT) på området snøsmelting ved bruk av sjøvann." [9].

Trondheim kommune, Trøndelag fylkeskommune og Bymiljøetaten i Oslo samarbeider om å finne en bærekraftig løsning for fjerning av forurenset snø i byene gjennom en konkurranse. Dette skal gjøres på en effektiv, skalerbar og fremtidsrettet måte, samt være klima- og miljøvennlig, og sirkulærøkonomisk [1]. Konkurransen involverer to separate aktører, der Rebel Garden, i samarbeid med AF gruppen og VisionTech, utgjør ett av de to konsortiene. Hovedidéen bak deres løsning er å utnytte temperaturforskjellen mellom sjø og snø for å smelte snøen på en mest mulig energieffektiv måte. Siden sjøvann som regel vil holde seg over 4 °C [10], vil det nesten alltid være noe energi som kan utnyttes til smelting av snøen. Dette gjør det mulig å plassere et anlegg i nærheten av byer, noe som reduserer transportavstanden. Ideen er kun aktuell for byer som ligger i nærheten av sjøvann.

Rebel Garden har allerede gjennomført fase 1 av snøsmeltingsprosjektet. Da ble det brukt poser med snø som ble flyttet fram og tilbake i sjøvann. Dette resulterte i at det dannet seg en klump med snø midt i posene, som deretter ble omgitt av kaldt vann og forhindret smelting av snøen. Dette har ført til behovet for forbedringer, og dermed er fase 2 av prosjektet planlagt, hvor denne bacheloroppgaven vil være involvert.

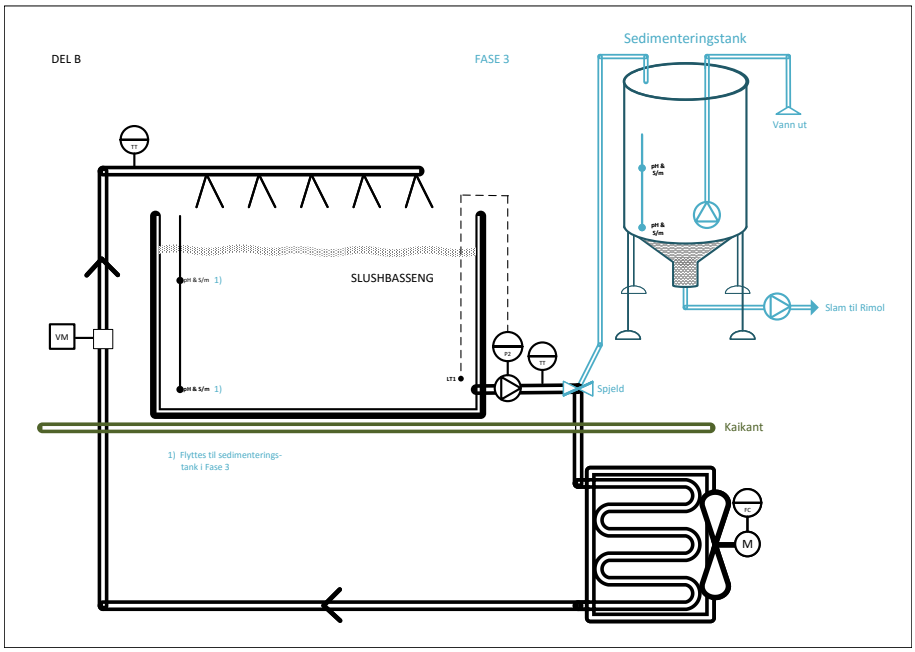
1.2 Oppgaveteksten

Snø fra bygatene skal smeltes på en miljø- og klimavennlig måte med minimalt energiforbruk. Idéen er å bruke temperaturdifferansen mellom snøtemperatur og sjøtemperatur til smelting gjennom en varmeveksler. Det skal stå et basseng på land der snøen skal dumpes. Deretter blir snøen pumpet gjennom en veksler som ligger i sjøen og opp i bassenget igjen. Det er i denne prosessen hvor snøen forhåpentligvis skal smelte. Etter at snøen er smeltet skal den tømmes over i en sedimenteringstank og stå til sedimenter har lagt seg. Anlegget skal være fullautomatisert, kun med manuell omstart ved ny påfylling av snø. Oversikt over anlegget vises i Figur 1.

Følgende oppgaver inngår:

1. Utvikle programvare for PLS:
   - Reguleringsalgoritmer
   - Datalogging
   - Styringslogikk med håndtering av motorer og sensorer, med alarmer og forriglinger
   - Logikk for analoge og digitale signaler
2. Utvikle programvare for HMI:
   - Prosessverdier
   - Trender
   - Alarmer
   - Systemkontroll
3. Designe og utvikle en IT-løsning for:
   - Datalogging i sky
   - Visualisering av data
4. Designe og bygge el-aut anlegg for del B
5. Forske på hvilke faktorer som påvirker smeltehastighet
6. Vurdere miljømessige aspekter

[11]



Figur 1: Konseptskisse over del B av anlegget

1.3 Problemstilling

Prosjektgruppen har som mål å utvikle et automatisert snøsmelteanlegg med skybasert overvåkning. Formålet med anlegget er å undersøke om snøsmelting ved bruk av sjøvann er en bærekraftig løsning, samt evaluere om det er mer miljø- og klimavennlig sammenlignet med dagens metoder. Anlegget skal gjennomgå partivise tester med snø, og resultater skal leveres innen fastsatte tidsfrister. Disse resultatene skal være relevante for videre bruk i SMELT-konkurransen. Prosjektet inkluderer også forskning på optimal styring av snøsmelteprosessen basert på data samlet inn fra systemet.

Målsetting: Utvikling av et automatisert snøsmelteanlegg med sjøvann for bærekraftig og miljøvennlig bysnøhåndtering.

1.4 FNs bærekraftsmål

Prosjektgruppen har valgt ut 5 av FNs 17 bærekraftsmål som vil være aktuelle for SMELT-prosjektet. Som nevnt tidligere er hensikten med SMELT-prosjektet å finne en miljø- og klimavennlig metode for å smelte snø. Dette innebærer å hindre at smeltevann fra deponier renner ut i natur, bekker og sjø, samt begrense transporten av brøytesnø. Figur 2 gir en oversikt over de aktuelle bærekraftsmålene. Prosjektet vil naturligvis berøre andre bærekraftsmål, men i mindre grad enn de fem presentert i dette kapittelet.

Figur 2: FNs bærekraftsmål [12]

Mål 6: Rent vann og gode sanitærforhold

Bærekraftsmål 6 innebærer at alle skal ha tilgang på rent vann og gode sanitærforhold. Ferskvann er en begrenset ressurs i verden, men FN mener at dersom ferskvannet forvaltes forsvalig, vil det være nok ferskvann til alle. Dette forutsetter at økonomi og dårlig infrastruktur ikke står i veien [12].

Når det kommer til hvordan dette bærekraftmålet er relevant i Norge og for prosjektet er det spesifikt delmål 6.3, 6.4 og 6.6 som er relevante.

Delmål 6.3: "Innen 2030 sørge for bedre vannkvalitet ved å redusere forurensning, avskaffe avfallsdumping og mest mulig begrense utslipp av farlige kjemikalier og materialer, halvere andelen ubehandlet spillvann og i vesentlig grad øke gjenvinning og trygg ombruk på verdensbasis [12]."

Delmål 6.4: "Innen 2030 betydelig bedre utnyttelsen av vann i alle sektorer og sikre bærekraftig uttak av og tilgang til ferskvann for å avhjelpe vannmangel og i vesentlig grad redusere antall personer som rammes av vannmangel [12]."

Delmål 6.6: "Innen 2020 verne og gjenopprette vannrelaterte økosystemer, inkludert fjell, skoger, våtmarker, elver, vannførende bergarter og innsjøer [12]."

> *I hovedsak har alle tilgang til rent vann til drikke, personlig hygiene og gode toalettforhold i Norge. Vannkvaliteten er god, men Norge har utfordringer knyttet til forurensing og rensing av avløpsvann. [...] Vi må gjøre mer for å beskytte våtmarksområder, myrer og økosystemer med ferskvann. I Norge er 35 prosent av innsjøer og elver forurenset eller skadet [12].*

Mål 11: Bærekraftige byer og lokalsamfunn

Bærekraftsmål 11 innebærer at en vil gjøre byer og lokalsamfunn inkluderende, trygge, robuste og bærekraftige. I dag bor over halvparten av verdens befolkning i byer, i tillegg til at byene står for 75% av verdens totale klimagassutslipp [12].

Spesifikt for prosjektet er delmål 11.6 den mest relevante.

Delmål 11.6: "Innen 2030 redusere byenes og lokalsamfunnenes negative påvirkning på miljøet (målt per innbygger), med særlig vekt på luftkvalitet og avfallshåndtering i offentlig eller privat regi [12]."

SMELT-prosjektet vil redusere byenes negative påvirkning på miljøet, dersom en får til å smelte snøen og rense smeltevannet fra miljøgifter.

Mål 13: Stoppe klimaendringene

Bærekraftsmål 13 går ut på å stoppe klimaendringene ved å begrense temperaturstigningen til godt under 2℃. Siden førindustriell tid har gjennomsnittstemperaturen på kloden steget 1℃, der størsteparten av oppvarmingen har skjedd siden 1980-tallet [12].

> *Det er viktig å begrense økningen av gjennomsnittstemperaturen til 1,5 °C dersom verden ønsker å slippe katastrofale konsekvenser i fremtiden [12].*

Spesifikt for prosjektet vil et smelteanlegg med sentral lokasjon redusere avstandene snøen må transporteres etter brøyting. Som nevnt i Kapittel 1.1, fraktes i dag brøytesnø til deponier langt utenfor byene. Med tung last og store kjøretøy medfølger det store CO_2 utslipp.

Mål 14: Livet i havet

Delmål 14.1: "Innen 2025 forhindre og i betydelig grad redusere alle former for havforurensning, særlig fra landbasert virksomhet, inkludert marin forsøpling og utslipp av næringssalter [12]."

Som nevnt i Kapittel 1.1 ble det i 2024 dumpet ut store mengder brøytesnø på sjøen i Larvik og Kragerø. Dette var på grunn av ekstreme snømengder som gjorde at kommunene ikke hadde kapasitet til å håndtere snømengdene. SMELT-prosjektet, dersom det når kapasitetsmålene, kan redusere sjansen for at hendelser som dette gjentar seg.

Mål 15: Livet på land

Bærekraftsmål 15 går ut på å "beskytte, gjenopprette og fremme bærekraftig bruk av økosystemer, sikre bærekraftig skogforvaltning, bekjempe ørkenspredning, stanse og reversere landforringelse samt stanse tap av artsmangfold [12]."

Spesifikt for prosjektet, er det som nevnt i Kapittel 1.1 et problem at snødeponier som smelter, slipper ut miljøgifter som er ødeleggende for økosystemer.

2 Teori

I dette kapittelet presenteres teori som legger grunnlaget for forståelse av rapporten. Teorien omhandler; smelteprosessen, reguleringsteknikk, PLS, HMI, IT/OT kommunikasjon og maskinvaren prosjektgruppen benyttet.

2.1 Smelteprosessen

For å få innsikt i hvordan en kan bruke sjøvann til å smelte snø, er det relevant å se på teorien knyttet til strømnings- og varmetransport. I varmetransport skiller en mellom de tre formene konduksjon, konveksjon og stråling. Likning 1 viser Fouriers varmelov for konduksjon i tre dimensjoner. Likning 2 viser konveksjon og Likning 3 viser Stefan-Boltzmanns lov for varmestråling. Ved å summere effektene fra de tre forskjellige varmetransportene kan en beregne sjøvannets smelteeffekt på snøen [13].

$$\vec{q} = -k \cdot \nabla T \tag{1}$$

$$q = \alpha \cdot \Delta T \tag{2}$$

$$q = \sigma(T_2^4 – T_1^4) \tag{3}$$

q	Varmeflux $[W/m^2]$
T	Temperatur $[K]$
k	Materialets termiske konduktivitet $[W/m - K]$
α	Varmeoverførings koeffisient $[W/m^2 - K]$
σ	Stefan-Boltzmann konstant $5.67x10^-8[W/m^2 - K^4]$

Ved konduksjon ledes varme gjennom aluminiumsrørets vegger på grunn av temperaturforskjellen mellom det varmere havvannet på utsiden og det kaldere vannet på innsiden.

Konveksjon beskriver varmeoverføringen mellom sjøvannets og rørets overflate, samt mellom rørets indre overflate og det kalde vannet inne i røret. Varmeoverføringskoeffisienten α påvirkes av sjøvannets hastighet rundt røret, samt slushens hastighet gjennom røret.

Varmestråling er en viktig del av teorien, men til vanlig spiller stråling en minimal rolle i varmeoverføring sammenlignet med konduksjon og konveksjon [13].

Et eksempel på hvordan en kan beregne effekten slushen smeltes med:

$$q_{\text{kond}} = -k \cdot \frac{\partial T}{\partial r}$$

$$Q_{\text{kond}} = \int_0^l \int_0^{2\pi} \int_{r_{\text{indre}}}^{r_{\text{ytre}}} q_{\text{kond}} \cdot r \, dr d\theta dz = \int_{r_{\text{indre}}}^{r_{\text{ytre}}} -k \frac{\partial T}{\partial r} \cdot 2\pi r l \, dr$$

$$Q_{\text{konv, sjø}} = \alpha_{\text{sjø}} \cdot A_{\text{ytre}} \cdot (T_{\text{sjø}} - T_{\text{ytre}})$$

$$Q_{\text{konv, slush}} = \alpha_{\text{slush}} \cdot A_{\text{indre}} \cdot (T_{\text{indre}} - T_{\text{slush}})$$

$$Q_{\text{total}} = Q_{\text{konv, sjø}} + Q_{\text{konv, slush}} + Q_{\text{kond}}$$

hvor $Q_{\text{konv, sjø}}$, $Q_{\text{konv, slush}}$, og Q_{kond} representerer de totale varmeoverføringene fra henholdsvis konveksjon på rørets ytre og indre overflate, samt konduksjon gjennom rørets vegg.

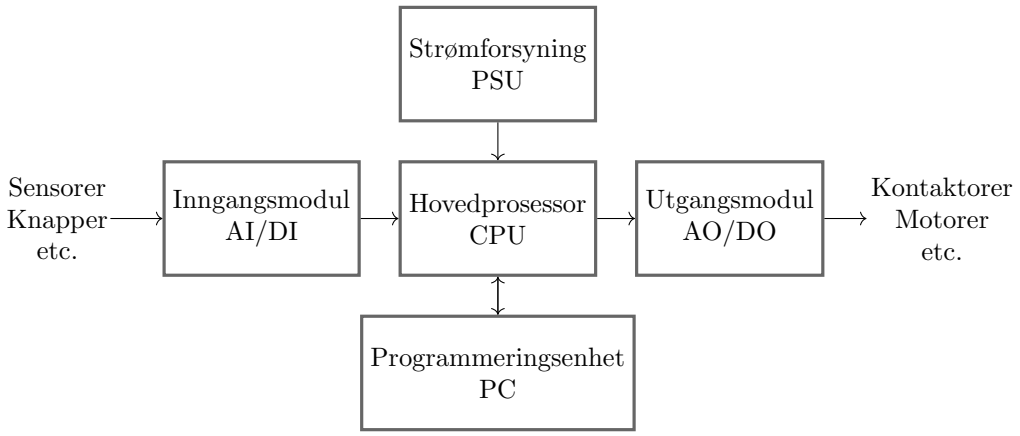
I prosjektet brukes ikke denne teorien direkte til å beregne smelteeffekt, da vi istedenfor gjør tester på anlegg og bruker sensordata til å beregne smeltehastigheten. Det er likevel nyttig å ha innblikk i hvilke tre måter varme transporteres på, samt hvilken påvirkning materialvalg, hastighet, og temperatur har på smelteeffekten. Senere i prosjektet bruker prosjektgruppen smeltehastighet i $[m^3/t]$, da dette er en mer anvendbar enhet enn smelteeffekt i $[w]$.

2.2 Maskinvare

Dette underkapittelet presenterer relevant teori for maskinvare benyttet under prosjektet.

PLS

PLS, programmerbar logisk styring, refererer til en enhet som gir mulighet for styring og overvåking av elektriske og automatiske systemer. Denne enheten representerer et alternativ til reléstyringer og byr på avanserte funksjoner for styring og overvåking [14]. En PLS er bygget for å fungere pålitelig under krevende forhold i industrielle sammenhenger, og kan inndeles i to hovedkategorier; Kompakt PLS og Modulær PLS. Kompakt PLS består av et gitt antall inngangs- og utgangsmoduler, mens en Modulær PLS består av en CPU modul som kan utvides med flere innganger og utganger [15].

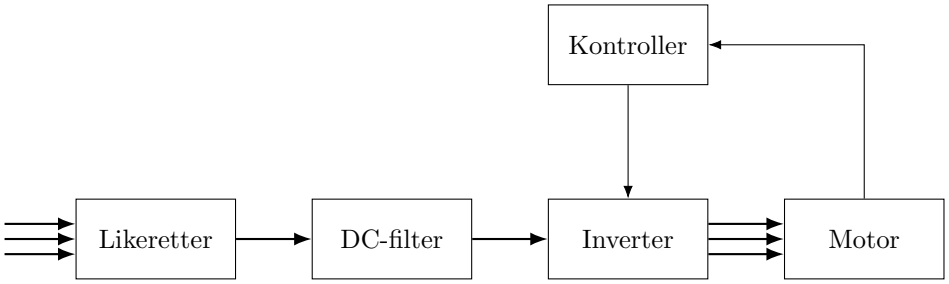


Figur 3: PLS struktur

Figur 3 viser en generell og overordnet struktur over hvordan en PLS er strukturert med hovedprosessor i sentrum.

Frekvensomformer

En frekvensomformer er en enhet som brukes til å regulere hastigheten til en elektrisk motor. Ved å variere frekvensen og spenningen til strømforsyningen som tilføres motoren, er det mulig å justere motorhastigheten. En av de viktigste egenskapene til frekvensomformere er evnen til å styre opp- og nedramping av motoren under start og stopp. Dette bidrar til å minimere belastningen på motoren og mekaniske komponenter, samt redusere støt og slitasje [16]. Dette bidrar også til system- og energieffektiv styring av motorer.



Figur 4: Frekvensomformer

Figur 4 illustrerer de fire hovedkomponentene som inngår i en frekvensomformer; likeretter, DC-filter, inverter og kontrollogikk.

Likeretteren omdanner trefase vekselstrøm (AC) til likestrøm (DC). Dette skjer ved hjelp av dioder eller tyristorer som lar strømmen passere i én retning for hver gang strømmen veksler. Dette steget gir en pulserende DC-spenning som må viderebehandles.

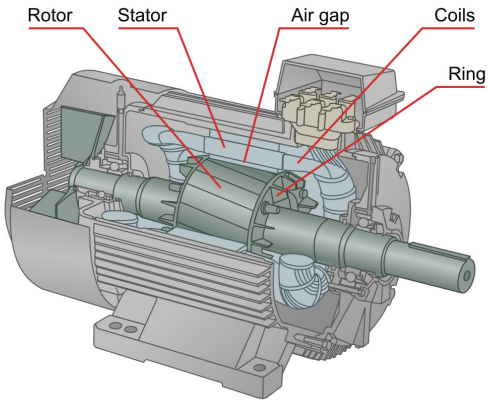
DC-filteret består vanligvis av kondensatorer og spoler som jevner ut den pulserende DC-spenningen fra likeretteren. Dette filteret fjerner også høyfrekvent støy og gir en ren og stabil DC-spenning. En jevn DC-spenning er viktig for å sikre drift av inverteren i neste steg.

Inverteren omdanner den filtrerte DC-spenningen tilbake til trefase AC, men nå med kontrollerbar frekvens og spenning. Dette gjøres ved hjelp av transistorer som styres av en kontrollogikk. Ved å variere frekvens og spenning på vekselstrømmen, kan inverteren kontrollere hastigheten til en elektrisk motor.

Kontrollogiken styrer hele frekvensomformeren og sikrer at likeretteren, DC-filteret og inverteren fungerer sømløst sammen. Den overvåker motorens turtall og regulerer transistorene i inverteren for å oppnå ønsket turtall, selv med varierende belastninger på motoren. [17]

Induksjonsmotor

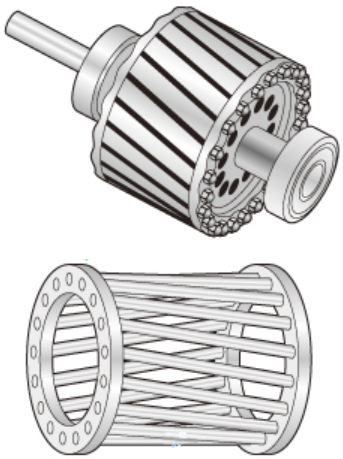
Blant elektriske maskiner er induksjonsmaskiner den markedsledende teknologien, med en markedsandel på 85% i industrien [18]. Maskiner som dette kan operere som både generatorer og motorer, alt ettersom om en vil omgjøre mekanisk energi til elektrisk energi, eller motsatt.



Figur 5: Illustrasjon av en induksjonsmotor [19]

En induksjonsmotor er bygget opp av to hovedkomponenter; en stator og en rotor. Statoren har en rekke viklinger med lav motstand gjerne fordelt på tre faser. Når det påføres vekselspenning på statorens viklingsterminaler, går det strøm gjennom viklingene og det utvikles et varierende magnetfelt. Måten statorvindingene er arrangert gjør at magnetfeltet vil rotere inne i motorhuset.

Rotoren består av en rekke tynne stenger som ligger litt skrått i forhold til parallellen til motoraksen. Ved endene av stengene er de kortsluttet med to metallringer. Rotoren og statoren er ikke i kontakt med hverandre, men har et minst mulig luftgap mellom, slik at den kan spinne fritt inne i motorhuset. Figur 6



Figur 6: Hele rotoren øverst, burvikling nederst [20]

Det magnetiske feltet som genereres i statoren, induserer en elektromotorisk kraft (EMF) i rotorens stenger. Dette fører til at det produseres en strøm i rotoren og et nytt magnetfelt induseres i rotoren med motsatt polaritet av det i statoren. Det roterende magnetfeltet i statoren vil deretter produsere et dreiemoment som trekker i feltet i rotoren og setter i gang en rotasjon.

Rotasjonshastighet

For å forstå hvordan en frekvensomformer kan variere hastigheten til en motor, må en se sammenhengen mellom frekvens, synkronhastighet og rotorhastighet. Synkronhastigheten beskriver magnetfeltets rotasjonshastighet og rotorhastigheten beskriver den mekaniske rotasjonshastigheten til rotoren.

I en induksjonsmotor vil rotorhastigheten alltid ligge litt etter synkronhastigheten. Differansen mellom hastigheten til magnetfeltet og rotoren (heretter kalt "slakk"), gjør at det induseres spenning i rotoren, som igjen får rotoren til å rotere.

Likning 4 beskriver forholdet mellom synkronhastigheten og frekvensen. Likning 5 beskriver forholdet mellom rotorhastigheten og synkronhastigheten, gitt at en vet slakken i motoren. Generelt ligger slakken på mellom 2% og 8%. Ut fra disse generelle likningene for induksjonsmotorer, kan en si at frekvensen er proporsjonal med den mekaniske rotasjonshastigheten.

$$n_s = \frac{120 \cdot f}{P} \tag{4}$$

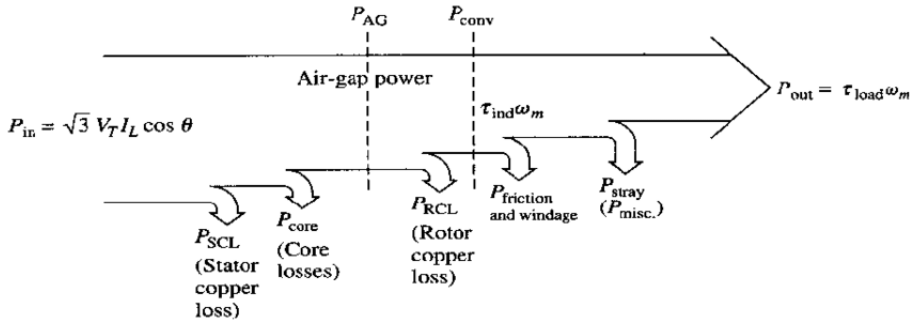
$$n_r = (1–S) \cdot n_s \tag{5}$$

Hvor:

n_s	Synkronhastighet (magnetfeltets rotasjonshastighet)
n_r	Rotasjonshastighet (Rotorens rotasjonshastighet)
S	Slakk i prosent
f	Frekvens
P	Antall polpar på statorvindingene

Dreiemoment & Effektbruk

Effektbruken til en induksjonsmotor avhenger av hvor mye mekanisk motstand den har på akslingen. En pumpe som går på 3000 rpm i luft vil ha betydelig mindre mekanisk last på akslingen enn en som går på samme hastighet i vann. Den vil derfor trenge mindre elektrisk energi. Figur 7 illustrerer sammenhengen mellom elektrisk energi og dreiemoment, inkludert effekttapene i motorhuset.



Figur 7: effektflytdiagram [21]

For å forenkle kan man samle alle effekttapene i P_{tap}, og uttrykke P_{inn} som i Likning 7.

$$P_{ut} = \tau_{last} \cdot \omega_m \tag{6}$$

$$P_{inn} = \tau_{last} \cdot \omega_m - P_{tap} \tag{7}$$

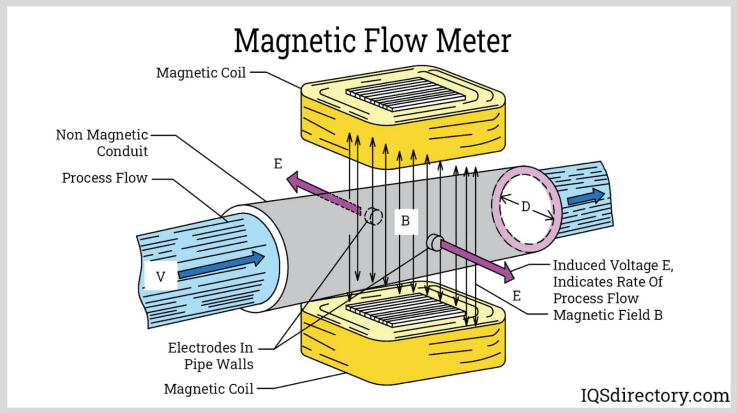
Motstandstermometer - RTD

Et motstandstermometer, også kalt RTD, er en type temperatursensor hvor den elektriske motstanden i metallet endrer seg proporsjonalt med temperaturendringen. Metallet har en gitt motstand ved en spesifikk temperatur, og dermed kan man måle temperaturen ved å måle motstanden i metallet.

I industrien er elementet PT-100 en mye brukt RTD. Elementet er navngitt etter metallet den består av, og den gitte motstanden. Pt er atomsymbolet for platina, og tallet 100 representerer at motstanden er $100\,\Omega$ ved $0\,^{\circ}\mathrm{C}$. Dette elementet brukes i flere typer sensorer, hvor de mest vanlige er vaierviklet og tynnfilmmotstand. Vaierviklet motstad er en trådkrets viklet rundt et keramisk element, og kan måle temperaturer mellom $-200\,^{\circ}\mathrm{C}$ og $600\,^{\circ}\mathrm{C}$. Denne typen motstand brukes når presisjon og stabilitet er viktig. Tynnfilmmotstand er et tynt lag platina på et substrat, og er mest egnet dersom det er nødvendig med rask responstid. Denne kan måle temperatur på mellom $-50\,^{\circ}\mathrm{C}$ og $250\,^{\circ}\mathrm{C}$. Tynnfilmmotstand har blitt mer vanlig å bruke de siste årene [22] [23].

Elektromagnetisk strømningsmåler

En elektromagnetisk strømningsmåler brukes for å måle strømning i et lukket rør, og består av en innenrørs spenningssensor og en en elektronisk transmitter. Sensoren har ingen bevegelige deler, og er derfor godt egnet til måling av strømning under høyt trykk uten fare for lekkasje.



Figur 8: Elektromagnetisk strømningsmåler prinsipp [24]

Måleren fungerer ved at det er plassert to spoler på hver side av et rør bestående av ikke-magnetisk materiale. Disse spolene danner et magnetfelt på tvers av røret. Videre er det plassert to elektroder på innsiden av røret, på tvers av magnetfeltet. Dette er illustrert i Figur 8. Sensoren baserer seg på prinsipper etter Faradays lov om induksjon som vist i Likning 8. Det stilles derfor krav til at den aktuelle væsken har en viss konduktivitet for å kunne indusere en målbar spenning, og varierer som oftest mellom 3 og 10 μS [24].

$$E = k * B * D * V \tag{8}$$

Hvor:

E	Indusert spenning
k	Proporsjonal konstant
B	Magnetisk feltstyrke
D	Distanse mellom elektroder
V	Hastighet til væske

Gitt en uniformt fordelt væske med konduktivitet mellom 3-10 μS som strømmer gjennom røret med hastighet V, vil det magnetiske feltet utføre et arbeid på de elektrisk ladede partiklene i væsken. Positivt og negativt ladde partikler vil trekke til hver sin side av røret og danne et spenningsfelt som plukkes opp av elektrodene i røret. Styrken på den induserte spenningen er proporsjonal med hastigheten til væsken. Transmitteren beregner hastigheten på væsken V ved å bruke den induserte spenningen E, styrken på avgitt magnetisk felt B, og distansen mellom elektrodene D, samt egne verdier for konstanten k avhengig av type væske.

2.3 Reguleringsteknikk

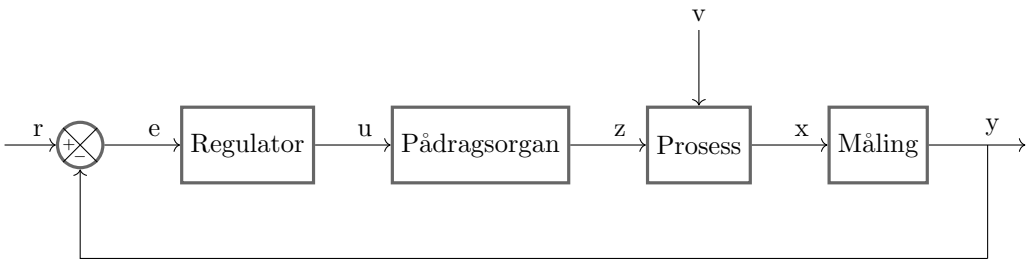
I de påfølgende underkapitlene presenteres relevant teori som tar for seg generell reguleringsteknikk, multivariable systemer, diskretisering, PID-regulering og empirisk prosessmodellering.

Generell reguleringsteknikk

For et automatiseringsprosjekt er det avgjørende med teori om reguleringsystemer, hva det innebærer og teknikker knyttet til dynamiske systemer.

Et reguleringssystem er et dynamisk og selvjusterende system som kan være utformet av mennesker for å oppfylle et behov. Det kan også være et system som har oppstått i naturen, for eksempel en biokjemisk prosess i menneskekroppen, som bukspyttkjertelen som regulerer blodsukker [25]. Reguleringsteknikk omfatter teknologiske prinsipper og metoder som benyttes for å automatisere eller styre maskiner eller prosesser, og brukes ofte synonymt med begrepet kybernetikk [26].

Sentralt for teori om reguleringsteknikk er prinsippet om tilbakekoblingssløyfer:



Figur 9: Tilbakekoblingssløyfe

Hvor:

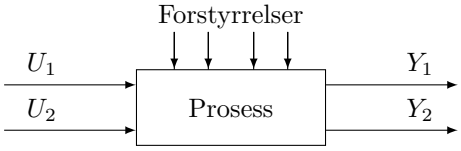
r	Referanse
e	Avvik
u	Pådrag
z	Utført pådrag
x	Tilstand
y	Måling
v	Forstyrrelse

Figur 9 er et enkelt blokkskjema som viser forholdet mellom de mest sentrale delene av et reguleringssystem. Prosessen har en tilstand x som måles med en sensor og blir utgangen y. Verdien av y blir tilbakekoblet og sammenlignet med referansen r, og det beregnes et avvik $e = r - y$. Avviket håndteres av regulatoren som avgir et pådrag u som virker på pådragsorganet, som igjen avgir et reelt pådrag z. Dette pådraget påvirker prosessen, som også blir påvirket av ytre forstyrrelser v, og igjen måles tilstanden. [25]

Multivariable systemer

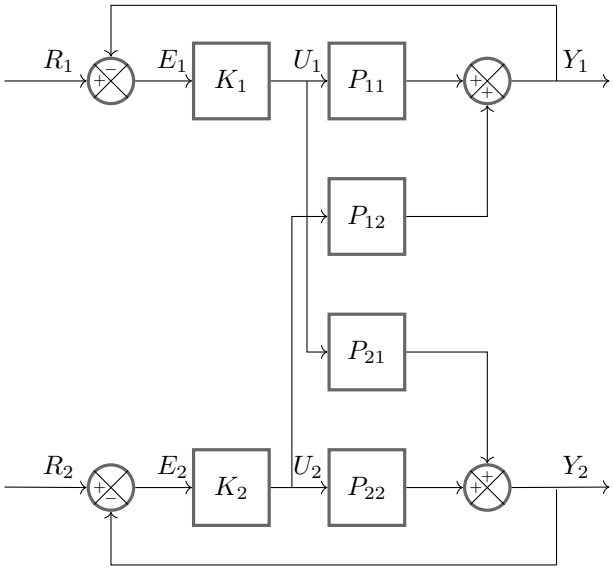
For utvikling av et anlegg som har som formål å smelte snø på en energieffektiv måte, er det flere variabler som har betydning og er av interesse. Det er derfor relevant å se på teorien bak multivariable systemer i et reguleringsteknisk perspektiv.

I et multivariabelt system oppstår det vekselvirkninger mellom de ulike kontrollerte og manipulerte variablene. Gitt et system med n ulike kontrollerte og n ulike manipulerte variabler eksisterer det $n!$ mulige kontrollkonfigurasjoner for det multivariable systemet [27].



Figur 10: Multivariabelt system (MIMO)

Figur 10 viser en enkel skisse av et multivariabelt system med like mange innganger og utganger, et tilfelle som kalles "kvadratisk". I de fleste tilfeller er det ønskelig med et kvadratisk system, slik at én inngang kan pares til én utgang, ovenfor et "ikke-kvadratisk" system, hvor antall inngangs- og utgangsvariabler er ulikt.



Figur 11: Vekselvirkninger i multivariabelt system

Figur 11 viser et eksempel på et multivariabelt system med to pådrag U_1 og U_2, og to målinger Y_1 og Y_2. For å karakterisere dynamikken til systemet tilstrekkelig, trengs det fire overføringsfunksjoner for prosessene som også inkluderer vekselvirkningene mellom inn- og utgangsvariablene. Disse kan løses med hensyn på vektoren for de kontrollerte variablene $\mathbf{Y(s)}$, og uttrykkes kompakt i matrise notasjon på formen:

$$\begin{bmatrix} Y_1(s) \\ Y_2(s) \end{bmatrix} = \begin{bmatrix} P_{11}(s) & P_{12}(s) \\ P_{21}(s) & P_{22}(s) \end{bmatrix} \begin{bmatrix} U_1(s) \\ U_2(s) \end{bmatrix} \tag{9}$$

Hvor prosessen P_{12} er vekselvirkningen av pådraget U_2 på utgangen Y_1, og prosessen P_{21} er vekselvirkningen av pådraget U_1 på utgangen Y_2.

Diskretisering

For regulering eller simulering av en prosess i et digitalt format, som ved i en datamaskin eller en PLS, behøves det å ta hensyn til tiden det tar for prosessoren å kjøre simuleringen, ettersom disse opererer i diskret tid. Det er derfor relevant å se på litt av teorien bak diskretisering av differensialligninger, over til det som kalles differensligninger.

Gitt en ulineær differensialligning på formen:

$$\frac{dy(t)}{dt} = f(y, u) \tag{10}$$

Hvor y er inngangsvariabel og u er utgangsvariabel. Likning 10 kan integreres numerisk gjennom en første ordens bakoverdifferanse hvor tiden t approksimeres til $t = kT_s$. Dette gir:

$$\frac{dy(t)}{dt} \tilde{=} \frac{y(k) - y(k-1)}{T_s} \tag{11}$$

Ved å sette sammen Likning 10 og Likning 11, og evaluere $f(y, u)$ for tidligere verdier av y og u; altså $y(k-1)$ og $u(k-1)$ får man:

$$\frac{y(k) - y(k-1)}{T_s} \tilde{=} f(y(k-1), u(k-1)) \tag{12}$$

Løser man for $y(k)$ får man en differensligning av første orden for utgangsvariabelen

$$y(k) = y(k-1) + T_s \cdot f(y(k-1), u(k-1)) \tag{13}$$

PID-regulering

Pumpen i anlegget må reguleres for å oppnå ønsket strømning. En PID-regulator vil være en god metode for å oppnå dette. Prosjektgruppen har brukt en innebygd funksjonsblokk fra TIA Portal. For å kunne bruke denne må en forklare grunnleggende teori om hvordan en PID-regulator fungerer.

En PID-regulator består av en proporsjonal-, integral- og derivatdel. Den matematiske sammenhengen mellom innsignal og utsignal for PID-regulatoren er:

$$u = K_p\left(e + \frac{1}{T_i}\int_0^t e\,dt + T_d\frac{de}{dt}\right) + u_0, \quad hvor\, e = r - y \tag{14}$$

Hvor:

r	Referanse
y	Målt prosessverdi (tilstandsverdi)
e	Avviket
u	Pådraget
u_0	Nominelt pådrag
K_p	Proporsjonal forsterkning (P-forsterkning)
T_i	Integrasjonstid
T_d	Derivasjonstid

Det resulterende pådraget u kan deles inn i separate pådrag fra hver del av regulatoren for henholdsvis P-, I- og D-delen, samt det nominelle pådraget.

$$u = u_P + u_I + u_D + u_0 \tag{15}$$

Hvor:

$$u_P = K_p \cdot e, \qquad u_I = \frac{K_p}{T_i} \int_0^t e dt, \qquad u_D = K_p \cdot T_d \frac{de}{dt} \tag{16}$$

P-delen Sørger for at pådraget endres proporsjonalt med avviket.
I-delen Fjerner det stasjonære avviket ved å integrere det over tid. Størrelsen på T_i avgjør hvor raskt integratorpådraget u_I gir like stort pådrag som P-delen.
D-delen Reduserer det dynamiske avviket og å motvirker endringer og oscillasjoner i prosessverdien. For høy verdi på T_d kan føre til ustabilitet [25].

PID_Compact V2

Prosjektgruppen benyttet TIA Portal sitt innebygde teknologiobjekt for PID-regulering, PID_Compact V2. Det er derfor relevant å presentere litt av den underliggende teorien for denne. PID_Compact er en brukervennlig ferdigløsning for en diskret PID-kontroller, og kan enkelt konfigureres til ønsket formål med automatisk finjustering av parametere. Videre tillater funksjonsblokken både inngangssignal direkte fra analoge innganger på PLSen gjennom parameteren "Input_PER", samt standard flyttall på parameteren "Input". På samme måte er det mulig å benytte tre former for utgangssignal; "Output_PER" som går direkte på analoge utganger med verdier mellom 0-10 V eller 4-20 mA, "Output" som må behandles etter behov i brukerprogrammet, og "Output_PWM" som gir et pulsbreddemodulert signal [28].

For regulering av en frekvensomformer er det i praksis gunstig med en PI-kontroller med relativt høy integrasjonstid. Likning 17 viser formelen for sprangrespons for PI-kontroller i PID_Compact V2.

$$y = GAIN \cdot X_W \cdot \left(1 + \frac{1}{TI \cdot t}\right) \tag{17}$$

Hvor:

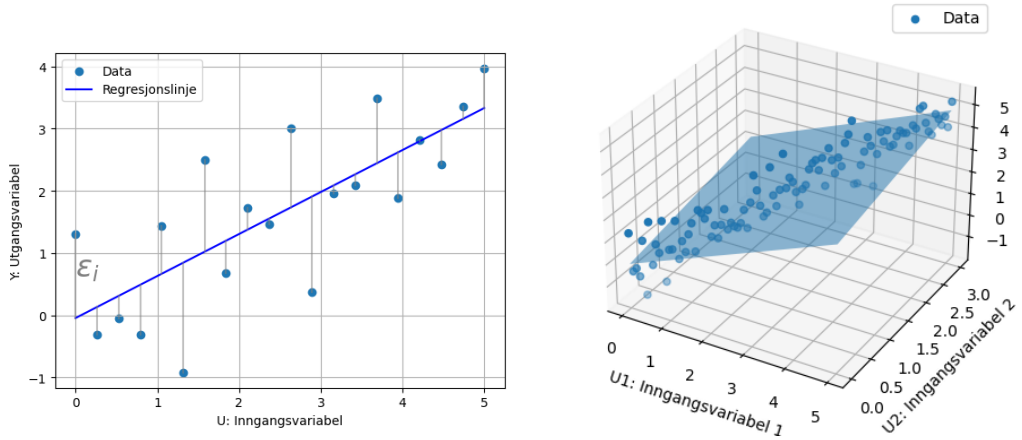
y	Utgangsverdi fra PID_Compact
GAIN	Proporsjonal forsterkning
X_W	Kontrollavvik
TI	Integrasjonstid
t	Tidsintervall siden sprang i kontrollavviket

Regresjonsanalyse og empirisk prosessmodellering

For beregninger på sensordata under prosjektet er det benyttet ulike metoder for regresjonsanalyse og prosessmodellering, og derfor presenteres underliggende teori for dette. For modellering av komplekse systemer hvor man enten ikke vet, eller det ikke er hensiktsmessig å utarbeide de underliggende dynamiske prosessene, kan man utarbeide en empirisk modell basert på data, gjennom en "svart boks tilnærming" [27]. Det finnes mange metoder for å lage en modell utifra empirisk data, men den vanligste metoden er lineær regresjonsanalyse vha. "minste kvadraters metode" [27].

Linær regresjonsanalyse

For å visualisere hvorvidt det er et lineært forhold mellom inngangs- og utgangsvariablene i datasettet kan man plotte disse opp mot hverandre, som vist i Figur 12. Her vises både for én inngangsvariabel i Figur 12a, samt for to inngangsvariabler i Figur 12b. Som illustrert i Figur 12b vil to inngangsvariabler gi et 3D-plot hvor regresjonsanalysen danner et plan. Dette impliserer at for flere enn to inngangsvariabler vil det være vanskelig å visualisere sammenhengen på tilsvarende måte.



(a) Enkel lineær regresjonsanalyse *(b) Flervariabel lineær regresjonsanalyse*

Figur 12: Lineær regresjonsanalyse

For enkelvariabel regresjonsanalyse kan hvert datapunkt i Figur 12a beskrives med den lineære funksjonen:

$$Y_i = \beta_1 + \beta_2 u_i + \epsilon_i \tag{18}$$

Hvor for hvert datapunkt i er; Y_i den faktiske målte verdien, u_i er inngangsvariabel og β_1 og β_2 er modellkoeffisientene man ønsker å estimere. Videre er restene $\epsilon_i = Y_i - \hat{y}_i$, hvor $\hat{y}_i$ er den predikerte verdien.

Likning 18 kan skrives på matriseform:

$$\begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_N \end{bmatrix} = \begin{bmatrix} 1 & u_1 \\ 1 & u_2 \\ \vdots & \vdots \\ 1 & u_N \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_N \end{bmatrix} \tag{19}$$

Hvor N er antall datapunkter. Videre kan Likning 19 skrives på kompakt notasjon:

$$\boldsymbol{Y} = \boldsymbol{U}\beta + \boldsymbol{\epsilon} \tag{20}$$

For å beregne koeffisientene β_1 og β_2 er det vanlig å benytte minste kvadraters metode, og minimere summen av de kvadrerte restene ϵ_i (SSE):

$$SSE = \sum_{i=1}^{N} \epsilon_i^2 = \begin{bmatrix} \epsilon_1 & \epsilon_2 & \dots & \epsilon_N \end{bmatrix} \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_N \end{bmatrix} = \boldsymbol{\epsilon^T \epsilon} \tag{21}$$

$$= (\boldsymbol{Y} - \boldsymbol{U}\beta)^{\boldsymbol{T}} (\boldsymbol{Y} - \boldsymbol{U}\beta) \tag{22}$$

Ved å løse Likning 22 for β får man [29]:

$$\boldsymbol{\beta} = (\boldsymbol{U^T U})^{-1} \boldsymbol{U^T Y} \tag{23}$$

Fra Likning 23 er $\boldsymbol{\beta} = \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_j \end{bmatrix}$ nå en vektor bestående av koeffisientene for j antall inngangsvariabler.

For regresjonsanalyse av to eller flere utgangsvariabler gjelder dermed også følgende:

$$\begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_N \end{bmatrix} = \begin{bmatrix} 1 & U_{11} & U_{12} & \dots & U_{1j} \\ 1 & u_{21} & U_{22} & & U_{2j} \\ \vdots & & & \ddots & \vdots \\ 1 & U_{N1} & U_{N2} & \dots & U_{Nj} \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \vdots \\ \beta_j \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_N \end{bmatrix} \tag{24}$$

$$Y_i = \beta_1 + U_{i1}\beta_2 + \dots + U_{ij}\beta_j + \epsilon_i \tag{25}$$

For å evaluere modellen er det vanlig å bruke metoden R kvadrert, $R^2 \in [0, 1]$:

$$R^2 = 1 - \frac{SSE}{SST} \tag{26}$$

Hvor SST er total sum av kvadratene [30].

ARIMA modellering

En annen robust metode for identifisering av dynamiske modeller basert på empirisk data er å bruke ARIMA, som er en kombinasjon av tre statistiske metoder: Autoregressiv (AR), integrert (I) og glidende gjennomsnitt (MA). ARIMA-modellen benytter tidligere verdier i en eller flere tidsserier av data for å kunne forutsi fremtidige verdier som en lineær kombinasjon av disse. Modellen tar inngangsparametrene $ARIMA(p, d, q)$, og kan uttrykkes generelt på formen [31]:

$$\left(1 - \sum_{i=1}^{p} \phi_i L^i\right)(1 - L)^d X_t = \left(1 + \sum_{i=1}^{q} \theta_i L^i\right)\epsilon_t \tag{27}$$

Hvor:

X_t	Tidsseriedata i tidssteg t
p	AR størrelseorden
d	I størrelsesorden
q	MA størrelseorden
ϕ_i	AR koeffisient
θ_i	MA koeffisient
ϵ_t	avvik i tidssteg t

L utgjør etterslep-operatoren ("lag operator"). Hvis det for eksempel er gitt en tidsserie $X = \{X_1, X_2, ...X_k\}$, så utgjør

$$L^2 X_t = X_{t-2}$$
$$L^{-3} X_t = X_{t+3}$$

Og generelt er $L^k X_t = X_{t-k}$ [32].

ARIMA er en videreutvikling av den mer kjente ARMA modellen, hvor forskjellen ligger i den integrerende delen (I) som benyttes dersom tidsseriedataen ikke er stasjonær. Ikke-stasjonær data innebærer gjerne data hvor gjennomsnitt, varians eller kovarians endrer seg over tid, og er i praksis uforutsigbart og vanskelig å modellere og forutsi [33]. Den differensierende delen d utgjør hvor mange ganger metoden y_t må differensieres for å bli stasjonær. En ARIMA-modell med null differensiering, $d = 0$, er i praksis en ARMA-modell [34].

2.4 PLS-programmering

Industrielle automatiseringsprosjekter bruker vanligvis PLS for å styre systemer. I dette prosjektet benyttes en PLS levert av Siemens, som krever bruk av deres eget programmeringsmiljø, TIA Portal, hvor både PLS og HMI kan programmeres i samme programvare. Først presenteres generell teori for PLS-programmering, og i påfølgende underkapitler presenteres teori spesifikt for Siemens programvare.

For å programmere en PLS er det mulig å bruke flere ulike språk/metoder. IEC 61131-3 er en internasjonal standard for PLS-programmering, som tar for seg fem måter å skrive et program på; ladderdiagram (LD), funksjonsblokkdiagram (FBD), strukturert tekst (ST), instruksjonsliste (IL) og sekvensdiagram (SFC). Ladderdiagram brukes oftest i forbindelse med relé og kontaktorlogikk og er brukervennlig. Funksjonsblokkdiagram er et program som er sammensatt av flere funksjonsblokker (FB), og bygger videre på digital elektronikk. Strukturert tekst er et høynivå programmeringsspråk som kan minne om C og Pascal. Her benyttes betingelser, løkker og funksjoner. Instruksjonsliste er et tekstbasert språk som ligner på assembler. Sekvensdiagram brukes for sekvensstyring, er visuelt intuitivt og ligner et flytskjema, med ulike trinn som henger sammen og styres av gitte overgangsbetingelser [35] [36] [37].

PLS tagger (PLC tags)

PLS tagger er variabler som benyttes i PLS-programmering. De inneholder data, som inn- og utgangsverdier, prosessvariabler og systemparametere. En tag tildeles en variabeltype og en adresse. Adressen må være unik, og ikke overskrive de innebygde reserverte adressene. Ved å organisere tagger i taglister er det lettere å feilsøke, sortere og holde orden i programmet [38].

Siemens PLS-programmering: TIA Portal

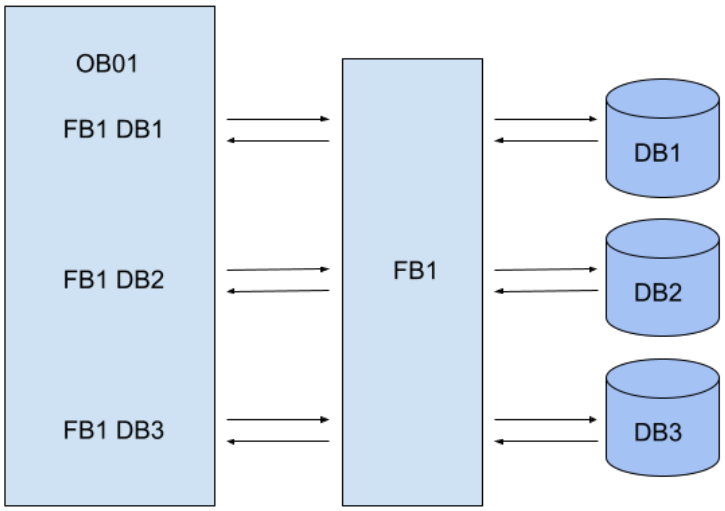
Strukturen inne i TIA Portal er bygget opp av Organisasjonsblokker (OB), Funksjonsblokker (FB), Funksjoner (FC) og Datablokker (DB). Man kan velge å skrive all kode inn i en Organisasjonsblokk, men sammen med de andre blokkene er det mulig å lage en mer organisert kode.

Organisasjonsblokkene (OB) er leddet mellom PLSens operativsystem og programkoden. De inneholder vanligvis lite kode, og brukes mest for organisering av FB og FC. En OB er hendelsessyrt, noe som betyr at en gitt hendelse får PLSen til å kjøre programmet som ligger i OB'en. Det finnes flere forhåndsbestemte OB'er, som blant annet kjøres basert på fysiske eller digitale avbrudd. OB1 er der hovedprogrammet kjøres, og kalles ofte for *Main*. Main OB1 har en syklisk oppførsel, som betyr at den kjører programmet den inneholder fra start til slutt, og begynner på nytt når den har kjørt gjennom hele koden [39].

Funksjoner (FC) er kodebiter som utfører en bestemt operasjon med gitte variabler. Det finnes mange ferdiglagde funksjoner i TIA Portal, men det er også mulig å lage egne funksjoner tilpasset eget program. Data som brukes i FC er kun lokal og midlertidig. Den blir dermed ikke lagret, med mindre den aktivt lagres som global variabel eller i en global datablokk. FC kan ikke direkte samhandle med inn- og utganger på PLSen, men funker godt til operasjoner som skal skje flere ganger i løpet av koden. Ofte brukes FC for å gjøre rene matematiske beregninger, og er dermed lettere å gjenbruke videre [40].

Funksjonsblokker (FB) er kodeblokker som samhandler med instanser fra en Datablokk, hvor variabler lagres. En FB kan kobles til flere Datablokker med lik struktur, og kan på den måten styre flere like enheter, med en og samme FB. Slike Funksjonsblokker og Datablokker brukes typisk for å styre flere motorer, eller lese verdier fra flere sensorer samtidig [41].

Datablokker (DB) er en måte å lagre og organisere data på. Datablokker kan dermed brukes for organisere variabler i større prosjekter, istedenfor PLS-tagger. Alle OB, FC og FB kan skrive data til en global DB, i motsetning til en intern DB hvor kun en gitt FB kan hente og lagre data. Datablokker med samme struktur, kan brukes i samme funksjonsblokk. Dersom flere objekter skal skaleres på samme måte, kan dette gjøres ved at hver av objektene har hver sin DB med like type inngangssignal, som benytter samme funksjonsblokk [42].



Figur 13: Sammenhengen mellom Operasjonsblokk, Funksjonsblokk og Datablokk

Alarm: For alarm- og feilsituasjoner finnes det ulike ferdigprogrammerte Organisasjonsblokker i TIA Portal som kjøres automatisk [43]. For detektering og diagnostikk av feilsituasjoner som dekkes av disse, kan det holde å sette en global variabel høy ved kjøring av den aktuelle OB'en. Eksempler på dette er organisasjonsblokkene:

Diagnostic error interrupt (OB82): Hvis den diagnosekompatible modulen oppdager en feil, avbryter OB82 den sykliske programbehandlingen.

Time error interrupt (OB80): Hvis den maksimale syklustiden overskrides, avbryter OB80 den sykliske programbehandlingen.

Pull or plug of modules (OB83): Operativsystemet på CPU-en kaller OB83 når en konfigurert og ikke-deaktivert modul eller undermodul i den distribuerte IO'en trekkes ut eller kobles til.

Rack or station failure (OB86): Når et DP-mastersystem, en slave eller en del av undermodulene svikter, kaller operativsystemet på CPU-en OB86.

Programming error (OB121): Operativsystemet i CPU-en kaller OB121 hvis det oppstår en programmeringsfeil under behandlingen av en instruksjon i brukerprogrammet.

IO access error (OB122): Operativsystemet i CPU-en kaller OB122 hvis det oppstår en feil under direkte tilgang til IO-data under behandling av en instruksjon i brukerprogrammet.

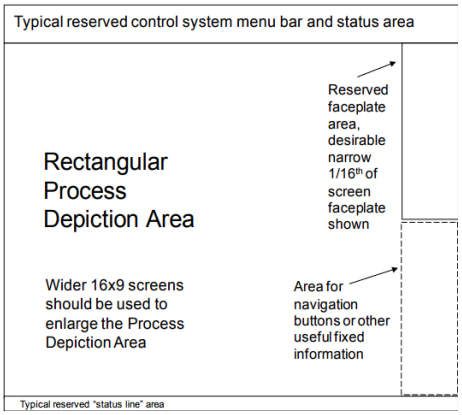
2.5 HMI-programmering

Brukergrensesnittet fungerer som et bindeledd mellom mennesker og maskiner, ofte referert til som HMI. Formålet er å gjøre kommunikasjonen enklere slik at brukere kan navigere gjennom systemet og få nødvendig informasjon. Innen IT refererer brukegrensesnitt til kontaktflaten mellom brukeren og et system, som for eksempel operativsystem, programmer og nettsider. [44].

Høy ytelse

Konseptet "Høy ytelse" handler om at designet i grensesnittet skal være minimalistisk. Fargebruken skal være begrenset til alarmer og viktige verdier. Ifølge "The high performance HMI handbook" [45] innebærer dette å skape et brukergrensesnitt som er behagelig å se på over tid, følger etablerte standarder og fremstiller rådata til intuitiv informasjon. Fargevalgene skal være konsistente og følge standard fargekoder fra dokumentasjonen i Figur 14. Industrielle standarder for fargebruk, indikatorer og layout gjør det mulig for operatører å navigere systemet effektivt, selv uten spesifikk opplæring. "The high performance HMI handbook" [45] påpeker at dårlige brukergrensesnitt har blant annet vært en medvirkende årsak til større ulykker flere ganger.

Color	RGB Values	Sample	Defined Uses
Gray	213, 213, 213		Overall graphic background
White	255, 255, 255		Highlighting of some small items, e.g., PV Quality indications
Light Gray	243, 243, 243		ON indication for equipment
Gray	136, 136, 136		Off indication for equipment
Dark Gray	74, 74, 74		Some text, minor process lines
Black	0, 0, 0		Text and labels, major process lines, process vessel outlines. Dark Gray (64, 64, 64) can also be a good choice.
Dark Blue	0, 0, 215		Process values, controller modes and outputs, similar special purposes. Trend line for a single trended value.
Dark Green	0, 128, 0		Controller setpoints and other operator inputs, trend trace of setpoints
Light Green	153, 255, 102		Possible "faint green" for some specific highlighting
Light Blue	187, 224, 227		Desired operating ranges or conditions
Cyan	0, 255, 255		Vessel level strips, trend lines
Brown	204, 102, 0		Trend lines, position feedback indiciation
Pale Red (Pink)	255, 153, 204		Possible "faint red" for some specific indications
Red	255, 0, 0		Top level, priority one alarm
Yellow	255, 255, 0		Priority two alarm
Orange	255, 102, 0		Priority three alarm
Magenta	255, 0, 255		Priority four alarm for diagnostics
Dark Magenta	204, 0, 102		Trend lines



Figur 14: High performance fargevalg og oppsett [46]

2.6 Operasjonell teknologi (OT) & informasjonsteknologi (IT)

Dette underkapittelet presenterer teori som bygger forståelse for operasjonell teknologi og informasjonsteknologi, samt hvordan disse teknologiene brukes til administrasjon, databehandling, kommunikasjon og til å styre og overvåke operasjonelle prosesser [47].

<table>
<tr><th>ISO/OSI</th><th>IT</th><th>OT</th></tr>
<tr><td>Applikasjonslag (5,6,7)</td><td>MQTT</td><td>PROFINET</td></tr>
<tr><td>Transportlag (4)</td><td>TCP</td><td rowspan="2">Samtid →
↓RT/IRT/TSN</td></tr>
<tr><td>Nettverkslag (3)</td><td>IP</td></tr>
<tr><td>Lenkelag (1,2)</td><td>LTE</td><td>Ethernet</td></tr>
</table>

Tabell 1: Forenklet OSI: IT/OT Protokollarkitektur

Tabell 1 viser en forenklet versjon av ISO/OSI modellen for sammensetningen av kommunikasjonsprotokollene som er relevant for dette prosjektet, hvor disse presenteres i mer detalj i Kapittel 2.6.1 og Kapittel 2.6.2.

2.6.1 OT-systemer

OT-systemer er programvare og maskinvare som interagerer med den fysiske verden, gjerne i industrisammenheng. Dette inkluderer industrielle styresystemer som PLS, HMI og drivere (f.eks en frekvensomformer) [48]. OT-systemer, som ofte er skreddersydde, har gjerne ansvaret for styringen av kritisk infrastruktur. Disse systemene må oppfylle strenge krav til pålitelighet og må kunne operere kontinuerlig døgnet rundt, med rask responstid [49].

Ethernet

Ethernet utgjør den mest tradisjonelle protokollen innenfor lokalnett (LAN), samt bredere datanettverk (WAN) som knytter sammen LAN over større geografiske områder. Protokollen opererer på lenkelaget (1 og 2) av ISO/OSI modellen, og er spesifisert i standarden IEEE 802.3 [50].

Datasendinger med Ethernet er strukturert i form av pakker og rammer. Hver datapakke som sendes består av rammer med binær nyttedata, samt tilleggsinformasjon som for eksempel EtherType og MAC-adresser [51]. MAC-adresse er den fysiske maskinvareadressen til enheter som kobles til et nettverk, og tilskrives som oftest en IP-adresse. EtherType indikerer hvilken protokoll som er innlemmet i datarammen, og forteller dermed mottaker hvordan informasjonen skal prosesseres. Eksempler på Ethertyper er IPv4, IPv6 og PROFINET [51].

For å koble sammen enhetene brukes Ethernet kabler, og disse finnes i mange varianter hvor man skiller mellom kategori, lengde, endepunkter, diameter på leder, og skjerming. Det finnes 8 ulike hovedkategorier (Cat1 - Cat8). Cat5e og Cat6 er blant de mest brukte Ethernetkablene, og støtter datahastigheter opptil 1 Gbps med henholdsvis 100 og 250 MHz båndbredde. Cat8.2 kan støtte datahastigheter opptil 40 Gbps med båndbredde på 2000 MHz [52]. Lederne i kabelen er vridd parvis, og kan enten bli skjermet med aluminiumsfolie, flettede metalltråder, eller begge. Videre

kan også selve kabelen skjermes med de samme materialene. Årsaken til vridde ledninger er for å utjevne elektromagnetisk interferens både fra utsiden, men også fra signalene i kablene [52].

PROFINET

PROFINET er en Ethernetbasert kommunikasjonsprotokoll på applikasjonslaget av ISO/OSI modellen, og er ledende industristandard for utveksling av data mellom styringsenheter som PLS og andre enheter i en automatiseringssetting [53]. PROFINET ansees som den mest avanserte og allsidige industrielle kommunikasjonsprotokollen, og er en videreutvikling av alternativet PROFIBUS som tidligere var markedsledende.

PROFINET bygger på standard Ethernet-protokoll og er kompatibel med vanlige Ethernetkabler, selv om bruk av industrigraderte kabler anbefales. Videre støttes kommunikasjon med andre Ethernetbaserte protokoller, som for eksempel OPC/UA og MQTT, og danner derfor en sømløs bro mellom OT og IT-systemer.

I likhet med andre Ethernet systemer, kan enheter i et PROFINET-nettverk kobles sammen gjennom en standard Ethernet-svitsj, så lenge denne støtter hastigheter på 100Mbps og simultan toveiskommunikasjon. Enheter i et PROFINET-nettverk kan også kommuniseres med trådløst gjennom protokoller som LTE, 5G, Wi-Fi eller Bluetooth.

For oppgaver som ikke er tidsavhengige eller har strenge krav til ytelse, som for eksempel sending av data fra en PLS til en sky, benyttes en standard TCP/IP kanal. For oppgaver hvor pålitelighet og høy ytelse i sanntid er viktig, går kommunikasjonsflyten direkte fra lag 2 til 7 av ISO/OSI modellen gjennom én av tre kanaler med følgende kvaliteter:

- RT - Deterministisk ytelse for automasjonssystemer i 1-10ms området
- IRT - Signalprioritering og planlagt svitsjing for presis synkronisering, ned til >1ms
- TSN - Under utvikling: Inkluderer båndbreddereservasjon og QoS

Alle kommunikasjonskanalene kan brukes samtidig, og ved deling av båndbredde garanteres det for at minst 50% av hver datasyklus forblir tilgjengelig for TCP/IP kommunikasjon [54] [55].

Standard Telegram 1

Standard Telegram 1 er et datautvekslingsformat for kommunikasjon mellom drivere over PROFINET og PROFIBUS. I prosjektets sammenheng er dette relevant for kommunikasjonen mellom PLS og frekvensomformeren.

Tabell 2 og 3 viser formatet til "Standard Telegram 1", og hvordan det brukes i sammenheng med frekvensomformere [56].

Kontrollord (STW1) og statusord (ZSW1) består av 16 bits hver, og er bekrevet i Tabell 4 og 5.

Adresse	**Navn**	**Innhold**
PZD 1	STW1	Kontrollord 1 (16bit)
PZD 2	NSOLL_A	Ønsket turtall (16bit)

Tabell 2: Sende Standard Telegram 1

Adresse	**Navn**	**Innhold**
PZD 1	ZSW1	Statusord 1 (16bit)
PZD 2	NIST_A	Faktisk turtall (16bit)

Tabell 3: Motta Standard Telegram 1

Hver bit i "Kontrollord 1" representerer en boolsk verdi som kan settes høy eller lav. Eksempel: "bit 3" som aktiverer drivverket dersom den settes høy, og deaktiverer dersom den settes lav [57].

På samme måte representerer hver bit i "Statusord 1" en boolsk verdi som settes høy eller lav av frekvensomformeren. Eksempel: "Bit 3" som settes høy dersom det er avvik på frekvensomformeren [57].

Tabell 4 og 5 viser betydningen av hver bit som "høy" og "lav". Ikke alle bitsene er med, da de ikke tas i bruk for motorstyring.

Bit	**Verdi**	**Betydning**
0	0	OFF1
0	1	ON
1	0	OFF2
1	1	Ingen OFF2
2	0	Hurtigstopp (OFF3)
2	1	Ingen hurtigstopp (OFF3)
3	0	Lås operasjon
3	1	Aktiver operasjon
4	0	Lås rampefunksjonsgenerator
4	1	Aktiver rampefunksjonsgenerator
5	0	Stopp rampefunksjonsgenerator
5	1	Fortsett rampefunksjonsgenerator
6	0	Lås settpunkt
6	1	Aktiver settpunkt
7	0	Ingen feilkvittering
7	1	Kvitter feil
10	0	Ingen kontroll av PLC
10	1	Hovedkontroll av PLC
11	0	Ingen retningsoinvert
11	1	Retningsoinvert

Tabell 4: Bits fra Statusord 1

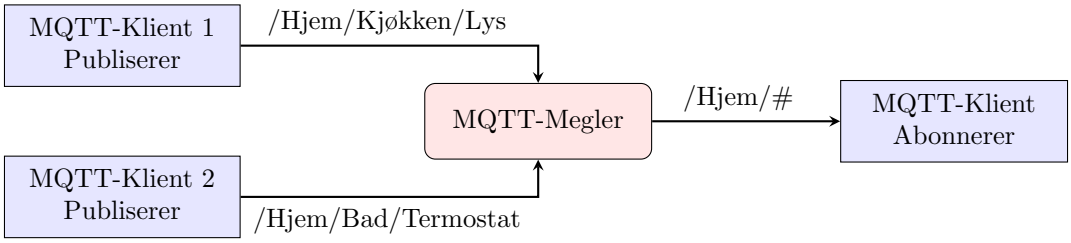
Bit	**Verdi**	**Betydning**
2	1	Aktiver operasjon
3	1	Feil aktiv
6	1	On-inhibit aktiv

Tabell 5: Bits fra kontrollord 1

2.6.2 IT-systemer

IT-systemer refereres til teknologien som behandler, lagrer, overfører og administrerer data. Slike systemer gjør det mulig for maskin og menneske å utveksle informasjon og data.

MQTT
Data fra en PLS kan logges til internett via en kommunikasjonsprotokoll. MQTT er en kommunikasjonsprotokoll som følger en publiser/abonner arkitektur hvor flere klienter kan publisere eller abonnere til ulike emner via en MQTT-megler. Megleren kan kjøres lokalt på en datamaskin, eller som skybasert løsning hvor klientene kobler seg opp på samme IP-adresse som megleren [58] [59]. Figur 15 viser et eksempel på hvordan klienter kan publisere og abonnere til emner hos megleren.



Figur 15: MQTT arkitektur

En MQTT-Klient kan blant annet være en PLS, en laptop eller en IoT-enhet som publiserer og/eller abonnerer på data fra et emne. For eksempel kan en termostat publisere temperaturdata til et emne kalt "/Hjem/Bad/Termostat", og en mobilenhet kan abonnere på emnet "/Hjem/Bad/Termostat". Eventuelt kan den abbonere på "/Hjem/#" for å motta pakker fra alle emner under "Hjem".

Tidsseriedatabase
Når en gjør målinger av snøsmelting er det nødvendig å ha en måte å lagre sensordata knyttet til et tidsstempel. På den måten kan en vite til hvilken tid en sensor målte en verdi. Her er det som regel aktuelt med en tidsseriedatabase.

En tidsseriedatabase er en database der målinger eller hendelser er knyttet til et tidsstempel. Dette er nyttig når en logger data som varierer over tid [60]. Eksempler på dette kan være værdata, aksjekurser, eller sensordata. Databaser driftes som regel på servere der en kan redigere databasen med SQL. Dette programmeringsspråket brukes til å etterspørre, endre og legge til data i databaser. I Kodesnutt 1 er de mest brukte SQL kommandoene listet opp [61].

TID	Nøkkel	Verdi
2024-01-30 23:59:59	Temperatur	6
2024-01-30 23:59:59	Fuktighet	44%
2024-01-30 23:59:59	Trykk	1003,5

Tabell 6: Tidsseriedatabase eksempel

```
SELECT -- ekstraherer data fra database
UPDATE -- oppdaterer data i en database
DELETE -- sletter data fra en database
INSERT INTO -- setter inn ny data i database
CREATE DATABASE -- lage ny database
ALTER DATABASE -- modifiserer databse
CREATE TABLE -- lager ny tabell
ALTER TABLE -- modifiserer tabell
DROP TABLE -- sletter tabell
CREATE INDEX -- lager ny index
DROP INDEX -- sletter index
```

Kodesnutt 1: SQL kommandoer

3 Metodikk

Dette kapittelet tar for seg hvordan prosjektgruppen har gått frem for å løse problemstillingen. Dette inkluderer fremgangsmåten under utvikling og testperiode, samt begrunning av valg av programvare og maskinvare for prosjektet.

3.1 Maskinvare

Dette underkapittelet tar for seg oppkobling og valg av maskinvare for anlegget. Tabell 7 gir en oversikt over maskinvare inni automasjonstavlen, og Tabell 8 gir en oversikt over maskinvare utenom;

Navn	**Modell / Beskrivelse**
PLS	ET 200SP
CPU	1512SP F-1 PN
Minnekort	Simatic S7, memory card
	Simatic HMI, memory card
Base enhet	BaseUnit BU15-P16+A0+2B
	BaseUnit BU15-P16+A0+2D
Input moduler	AI 4xRTD/TC HF
	AI 4xI 2-/4-wire ST
	AI 4xI 2-/4-wire ST
Output moduler	DO 8x24VDC/0.5A ST
HMI	TP 700 Comfort
Frekvensomformer	Sinamics G120C
FQ-panel	Sinamics G120 BOP-2
Strømforsyning	Omron 400V/24V 5A
Ethernet-svitsj	JetNet 5010G
IoT-gateway	Teltonika TRB 140
Transmittere	FT01: SITRANS FM MAG 6000
	QT01: JUMO ecoTRANS pH 03
	CT01: JUMO ecoTRANS Lf03
Kontaktor	3-pole 1NO+1NC, 25A
Motorvern	Motorvern 20-25A GV2ME22
Automatsikring	IC60H 2P 06A/B
	IC60H 3P 32A/C

Tabell 7: Utstyrsliste: Tavle

Navn	**Modell / Beskrivelse**
Sensorer	TT01: RND 410-00281
	TT02: RND 410-00291
	TT03: RND 410-00329
Motor/pumpe	KTZ411-53 11kW 400v
Konteiner	12 fots konteiner
Veksler	Aluminium
Båt	21 fot tauebåt

Tabell 8: Utstyrsliste: utenfor skap

PLS modulsystem

PLS løsningen valgt for autoanlegget er et EP200SP modulsystem som bygger seg rundt en CPU i Siemens S7-1500 familien, med mulighet for et stort utvalg tillegsmoduler. I dette prosjektet tas det i bruk en CPU, basemoduler, analoge inngangsmoduler, RTD inngangsmoduler og en digital utgangsmodul. Valg av Inngang-/utgangsmoduler gjøres basert på instrumentene en ønsker å koble opp.

CPUen sammen med alle tillegsmodulene monteres samlet på en DIN-skinne. Det behøves ingen verktøy for å installere modulene, ettersom disse klikkes på vha. en mekanisme.

Sensorer

For temperatursensorer er det tatt i bruk RTD inngangsmoduler, som måler temperatur ved hjelp av motstanden til metallet i sensoren. Denne teorien er gått gjennom i Kapittel 2. For sensorer som har strømforsyning ved hjelp av en transmitter, tar prosjektgruppen i bruk analoge inngangsmoduler laget for å ta i mot $4 - 20mA$ signaler. I prosjektets tilfelle gjelder dette transmitterene for strømning, pH-måling og vannkvalitet.

Alle tre temperatursensorer er av typen PT100 og konfigureres på samme måte i TIA-portal. Her må konfigurasjonen testes og kryssjekkes mot en selvstendig temperaturmåler for å sjekke om verdiene stemmer.

Transmitterene for strøming, pH-måling og vannkvalitet gir alle et $4 - 20mA$ strømsignal til en Analog inngangsmodul, men må kalibreres basert på oppkoblet sensor.

For strømings sensoren i røret settes det en maksverdi på 100 m^3/t som da tilsvarer et signal på $20mA$. På denne måten kan man kalkulere i TIA-portal at maks analog verdi tilsvarer 100 m^3/t og minimum analog verdi tilsvarer 0 m^3/t.

pH- og vannkvalitetstransmitteren kalibreres ved å bruke en referansevæske med kjent verdi. Når sensoren har et referansepunkt kan den utifra dette beregne ønskede målinger.

Frekvensomformer: SINAMICS G120C

SINAMICS G120C har et kompakt design, integrerte sikkerhetsfunksjoner og andre funksjoner for ulike bruksområder. Den er konstruert for å kunne installeres i elektriske anlegg og maskiner. Frekvensomformeren som blir brukt benytter både PROFINET og Ethernet/IP. Dette er viktig for å kunne styre frekvensomformeren fra PLSen. Merkeeffekten er 15 kW og merkestrømmen er 31 A som betyr at den er litt overdimensjonert i forhold til pumpen som blir brukt.

Kommunikasjon mellom maskinvareenheter

HMI, frekvensomformer og PLSen kommuniserer gjennom PROFINET og er alle koblet til en felles Ethernet-svitsj. På denne måten settes hver maskinvareenhet til en unik IP-adresse og kan kommunisere. Dette lokale nettverket utgjør OT-systemet i anlegget. Teorien bak OT-systemer er gjort rede for i Kapittel 2.6.1.

Fordelen med å styre frekvensomformeren over PROFINET er at en slipper å instrumentere for; fem digitale inngangsporter, to digitale utgangsporter og én analog utgangsport. Over Ethernet brukes strukturen "Standard Telegram 1"; en 16-bits struktur for å kommunisere i sanntid med frekvensomformeren.

Kontaktor, motorvern, automatsikring og strømforsyning

Kontaktoren "3-pole 1NO+1NC, 25A" fra Siemens er en 3-polet kontaktor som i prosjektets tilfelle brukes som en bryter på motorkretsen. Bryteren håndterer tre faser som er essensielt for en 3-fase motorkrets. Bryteren har mulighet til å starte og stoppe en krets ved hjelp av et 24 V likespenningssignal fra en styreenhet. I prosjektets tilfelle styres kontaktoren gjennom en holdekrets i programvaren på PLSen.

"IC60H 3P 32 A/C" er en 3-polet automatsikring med 32 A merkestrøm og utløserkarakteristikk C. Denne automatsikringen står først i skapet, og beskytter kretsen fra skapet ut til motoren. Utløserkarakteristikken C, er valgt på bakgrunn av at en ønsker en langsommere reaksjonstid når en dimensjonerer en motorkrets. Dette gjør at motorene kan trekke litt overstrøm ved oppstart.

"IC60H 2P 06A/B" er en 2-polet automatsikring med 6 A merkestrøm og utløserkarakteristikk B. Denne automatsikringen står etter den 3-polede automatsikringen og før strømningstransmitteren som bruker 230 V. Utløserkarakteristikken B er valgt på bakgrunn av at B er en mye brukt karakteristikk som gir middels hurtig respons.

Motorvernet er dimensjonert på bakgrunn av pumpens merkestrøm på 22 A. "Schneider Electric GV2ME22" er en 3-polet, termisk-magnetisk motorvernbryter. Den termiske tripperen hindrer at motoren blir overbelastet, men vil ikke reagere på korte transienter med overstrøm. Den magnetiske tripperen slår inn på kortslutninger i kretsen, altså plutselige store endringer i strømmen. Den magnetiske delen kan slå ut i tilfeller der strømmen er opp imot $13x$ merkestrømmen.

Strømforsyningen "Omron 400V/24V 5A" tar inn 400 V vekselstrøm som transformeres og likerettes ned til 24 $V(DC)$. PLS-modulene, Teltonika-enheten og transmitterene for pH og vannkvalitet, tar i bruk 24 $V(DC)$.

Pumpe

"KTZ411-53 11kW 400V" er en 3-fase lensepumpe. Denne pumpen er spesifikt designet for å være robust og tåle grovere vannmasser. Den har blant annet evnen til å tørrkjøres som gjør den pålitelig under krevende forhold. Spesifikasjonene til pumpen er noe utfordrene, da den har en startstrøm på $152,9$ A sammenliknet med merkestrømmen på 22 A. En startsrøm på $7x$ merkestrømmen krever at en må rampe opp turtallet til motoren langsomt for å hindre stor startstrøm. Alternativt kan man dimensjonere trege automatsikringer og motorvern, som håndterer høye startstrømmer.

Båt, konteiner og veksler

For å forflytte sjøvannet rundt veksleren ble det leid inn en 21 fot taubåt fra Trondheim havn. Denne hadde en ukjent motorkraft, så sjøstrømning måtte dermed regnes utifra turtallet. Båten ble fortøyd i kaikanten slik at kjølvannet drev forbi veksleren. Konteineren var på 12 fot. Denne ble fylt med to betongklosser på 1 m^3 for å minimere volumet for testing. Veksleren var et aluminiums rør med diameter på 250 mm med tykkelse på 7 mm utenom bendene som er 6 mm. Figur 23 fra Kapittel 4.1 viser et oversiktsbilde.

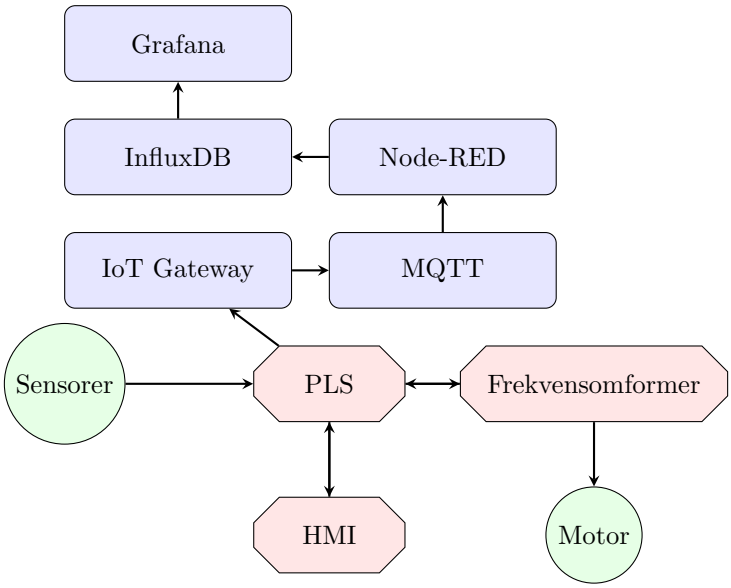
3.2 Systemarkitektur

Dette underkapittelet tar for seg programvarearkitektur. Det gås gjennom hvilke metoder som er brukt for OT- og IT- systemet.

OT-systemet består av PLSen, frekvensomformeren og HMIen som alle kommuniserer gjennom PROFINET.

IT-systemet består av datalogging og alt av skybaserte tjenester. Dataloggingen er utviklet slik at PLSen logger prosessdata til den skybasert tidsseriedatabasen InfluxDB, og deretter gir muligheten til å visualisere dataene i en hvilken som helst nettleser gjennom Grafana.

Dette underkapittelet går gjennom metodene brukt for strukturere systemet. Figur 16 gir en oversikt over hele arkitekturen.



Figur 16: Systemarkitektur

3.2.1 PLS

Til programmering av PLS og HMI er det benyttet utviklingsplattformen TIA Portal V18. Et viktig og gjennomgående prinsipp for programvareutvikling er skalerbarhet. Dette innebærer: ryddighet, standardisering, objektorientering, bruk av engelske variabelnavn, samt hierarkisk program- og variabelstruktur.

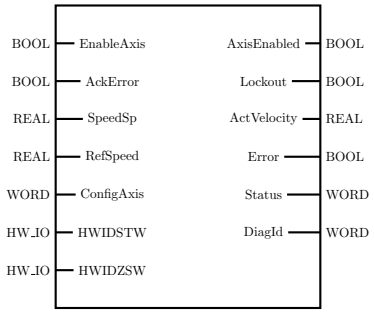
Programkoden struktureres i egne mapper etter hovedoppgaver (som alarm, regulering eller kommunikasjon). Alle hovedoppgavene i programmet kjøres i en OB. Hovedoppgavene er laget som Funksjoner (FC). Inni Funksjonene (FC) kan det ligge gjenbrukbare Funksjonsblokker (FB) og annen kode for å utføre hovedoppgaven. Videre benyttes Datablokker (DB) i størst mulig grad for lagring av variabler fremfor bruk av PLS Tagger, ettersom Datablokker er mer allsidig og benytter dynamisk minneallokering. Dette kan utelukke fare for overflyt mellom to variabler dersom disse er feilallokert og overlapper. For standardiserte og gjenbrukbare variabler, som for eksempel for variabler som gjelder for en motor, benyttes det brukerdefinerte typer (UDT). Brukerdefinerte typer støtter prinsippet om objektorientert kode og følger en hierarkisk variabelstruktur hvor man kan ha flere undergrupper av variabler som gjelder generelt for samme type objekt.

Motorstyring

For styring av frekvensomformeren skrives det programvare på PLSen som sender og mottar datapakker. Siemens har ferdigprogrammerte funksjonsblokker som benytter seg av “Standard Telegram

1", i tillegg kan man også programmere egne funksjonsblokker som tar i bruk andre datautvekslingsformater som; "Siemens Telegram 111" og "PROFIsafe telegram".

Metoden prosjektgruppa har valgt å benytte seg av er "SINA_SPEED"; en brukervennlig funksjonsblokk utviklet spesielt for Siemens frekvensomformere. Figur 17 viser innganger og utganger, samt datatypene blokken bruker.



Figur 17: SINA_SPEED blokk [57]

3.2.2 HMI

HMI-panelet er av typen SIMATIC TP-700 Comfort panel. Skjermen er 7"og er plassert på utsiden av døren på automasjonsskapet, som har dimensjonene 800x800x300mm. Panelet programmeres i Siemens TIA portal V18, samme som PLSen.

I prosjektet som opprettes i TIA Portal konfigureres type panel som skal brukes. Det er mulig å legge til flere typer paneler; Comfort, Mobile og IPC477E [62]. TIA Portal har innebygde funksjoner som gjør at en kan lage et oversiktlig og ryddig HMI-program. Dette innebærer funksjonalitet som maler, sprett-opp-vinduer, innfallsmenyer og skript. For å få HMIen til å samhandle med resten av systemet må en opprette tagger under "HMI taglist" og knytte disse opp mot PLS-programmet.

høy ytelse er et viktig prinsipp innenfor utvikling av brukergrensesnitt. Siden oppdragsgiver har hatt få krav til hva som skulle inngå i brukergrensesnittet, har høy ytelse-standarder spilt en viktig rolle for utseende og funksjonalitet i HMIen. Dette gjelder valg av farger, grafikk, oppsett, ikoner og skrift.

Alarmsystem

Det er viktig at avvikssituasjoner på anlegget oppdages og varsles. For å oppnå optimal varsling, ble det kartlagt mulige avvikssituasjoner. Dette innebærer avvik i kommunikasjon, avvikende sensorverdier, feil på hardware, uønskede prosessverdier og systemfeil. Ved noen tilfeller kan PLS-taggene konfigureres rett inn i HMI-alarms som "trigger tagger" for avvikssituasjoner, slik som sensorverdier. For andre typer avvik er det nødvendig med egne kodesnutter. Det ble dermed laget en funksjon, "Alarm [FC2]", inne i PLS-programmet, som inneholder kode for å overvåke og definere ulike avviksituasjoner. Alamene ble deretter konfigurert med trigger taggen fra funksjonen i HMI-alarms.

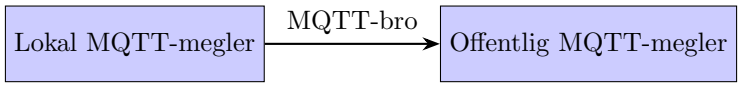
3.2.3 IoT Gateway

For å kunne loggføre sensorverdier fra PLSen til databasen i sanntid blir det brukt en Teltonika TRB 140 for trådløs LTE/4G tilkobling. Enheten er koblet til PLSen med en Ethernetkabel, slik at de er på samme lokale nett.

Teltonika TRB 140 kan konfigureres ved å logge inn på nettgrensesnittet på lik måte som en ville gjort med en tradisjonell WiFi-ruter. Inne på nettgrensesnittet, typisk tilgjengelig på "192.168.0.1", kan en konfigurere en MQTT-megler, som da driftes på enheten.

3.2.4 MQTT

MQTT oppsettet er satt opp ved hjelp av en lokal og en offentlig MQTT-megler. Den lokale er driftet på Teltonika-enheten og den offentlige av HiveMQ. Mellom meglerne er det en MQTT-bro som sørger for at meldinger blir publisert til den lokale MQTT-megleren og videresendt til den offentlige MQTT-megleren. Det er mulig å gjøre dette bidireksjonelt, men siden systemet kun skal kommunisere i en retning, er det bare den lokale som viderepubliserer til den offentlige.



Figur 18

Som gjort rede for i teorien publiserer og abonnerer enheter til emner. I prosjektets tilfelle publiseres meldingspakker fra PLSen til emnet "anlegg/del_b". Dette skal være et emne som representerer del B av anlegget. Dersom en ønsker å skalere anlegget, og logge data fra del A, eller eventuelt flere deler av anlegget kan man opprette nye emner som "anlegg/del_a" og "anlegg/del_c".

Siden emnene er bygget opp i et hierarki, kan en MQTT-klient abonnere til "anlegg/" og da motta meldinger fra alle delene av anlegget. Dette egner seg godt dersom anlegget skal skaleres opp.

3.2.5 Node-RED

Node-RED brukes som en abonnent hos den offentlige MQTT-megleren. Node-RED abonnerer på emnet "anlegg/del_b" og mottar dermed meldigspakkene fra PLSen. Disse meldingspakkene kommer inn som JSON-strenger, som er nødt til å behandles slik at dataen kan sorteres og sendes videre til tidsseriedatabasen InfluxDB.

Node-RED fungerer godt som en oversiktlig metode for å sortere og plassere data hentet ut fra en MQTT-megler. Den visuelle måten å kode på i Node-RED gjør det også veldig intuitivt å forstå dataflyten. Tjenesten kan også brukes til å feilsøke meldingspakkene før de når databasen.

3.2.6 InfluxDB

InfluxDB er en tidsseriedatabase som kan hente og lagre store mengder data. Målinger fra sensorene hentes via Node-RED og lagres i en felles database som individuelle målinger i InfluxDB. Ved å velge en database og en måling i datautforskeren genereres en kodesnutt i SQL for å få tak i den spesifikke målingen.

3.2.7 Grafana

Grafana er en åpen kildekode plattform for visualisering og overvåking av data. Programmet er nettleserbasert og dermed kan hvem som helst med gitt tilgang gå inn og overvåke systemet til enhver tid. Oppdragsgiver ønsket dermed at prosjektgruppen skulle benytte Grafana for visualisering.

Data som skal visualiseres hentes fra InfluxDB gjennom å legge til en valgt *Bucket* fra InfluxDB, som en *Data source* (kilde til database) i Grafana. På den måten lages det et *Dashboard* (panel) i Grafana hvor det kan legges til grafer, tabeller og diagrammer for å visualisere data på egnet måte. For å visualisere ønsket måling i en graf, brukes identisk SQL-kode som ble generert i InfluxDB for den gitte målingen. Målinger med samme måleenhet, som temperatur, ble lagt inn i samme graf.

Grafana har en innebygget funksjon hvor rapporter med data sendes automatisk ved valgte tidsintervaller. Oppdragsgiver ønsket rapporter for hvert parti, og ikke for et tidsintervall. Dermed ble det laget et eget Python skript som sender rapporter med data fra hvert enkelt parti med

snø. Skriptet kjøres på AWS med funksjonen “Elastic Beanstalk”, som gjør det lett å integrere kodeskript inn i Grafana. Den tar i bruk Grafana sitt http API, for å etterspørre rapporter fra det valgte panelet. Skriptet kjører hele tiden, og når et nytt parti starter, blir rapport fra forrige parti sendt til en valgt e-postadresse.

3.3 Fremgangsmåte for testperiode

Hele anlegget skulle testes i andre halvdel av mars, organisert av oppdragsgiver fra Rebel Garden. Målet med testperioden var å teste om konseptet fungerte slik som ønsket, og å skaffe data for å gjøre beregninger på smeltekapasitet.

Anlegget ble plassert på kaien til Betong Øst i Fagervika i Trondheim. Det besto av en varmeveksler, kranbil, konteiner/basseng til snø, motorbåt, strømaggregat og en liten brakke til kontor, automasjonsskap og annet utstyr. I testperioden settes alle disse delene sammen, også med program som bachelorgruppen har utviklet. Under er en oversikt over hendelseforløpet i testperioden.

DAG 1, 13.mars: Befaring på anlegget og planlegging for dagene fremover.

DAG 2, 14. mars: Monterte opp skapet i arbeidsbrakken. Koblet opp all resterende instrumentering inn i skapet og plasserte sensorene på egnet plass. Lufttemperatur på baksiden av arbeidsbrakken i le for vind. Strømingsmåler ble festet rett ved utgangen av konteineren. Temperaturmåleren for returtemperaturen ble festet i ytterkant av røret hvor vannet kommer inn i konteineren igjen. Temperaturmåler for sjø ble festet slik at den lå flytende i vannkanten ved veksler. Pumpen ble senket ned i sjøen med rørforbindelse til konteineren for å fylle den med vann. Dette ble gjort for å se at alt fungerte før snølass ble tilgjengelig.

DAG 3, 15. mars: Pumpen ble montert i konteineren. Alle rørforbindelser mellom konteiner, strømingsmåler og veksler ble montert i samarbeid med kranbilfører. Pumpen ble forsøkt styrt med frekvensomformeren uten at dette fungerte.

DAG 4, 18. mars: Fungerte ikke å styre pumpe med frekvensomformer. Mye tid til feilsøking. Koblet ut koblingsboks mellom pumpe og frekvensomformer. Viste seg at feilen lå her.

DAG 5, 19.mars: Veksleren ble fraktet ut og holdt i vannet ved hjelp av kranbil. Anlegget kjørte første parti med snø. Pumpen ble styrt manuelt fra TIA Control Panel. Pådrag på pumpen ble satt til maks (2935 rpm).

DAG 6, 20. mars: Parti 2 og 3 med snø ble gjennomført. AWS prøveperiode var utgått, og ny MQTT løsning fungerte ikke optimalt. Dermed ble målinger registrert og loggført via skjermbilder fra innebygd trendovervåking i TIA Portal. Smeltet snø fra konteineren ble pumpet med liten pumpe opp i sedimenteringstank.

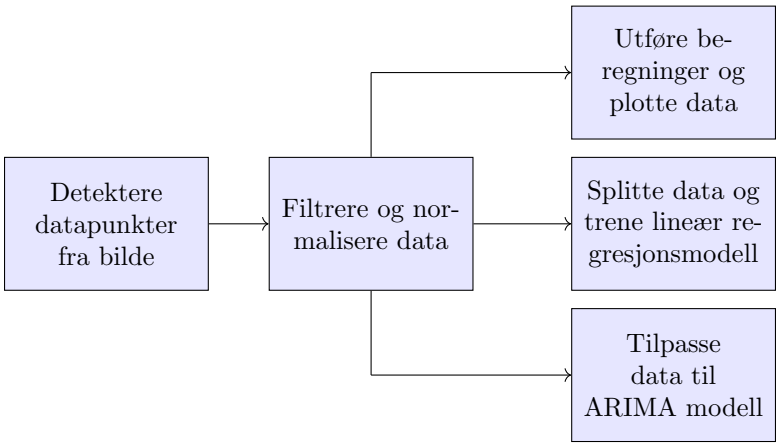
DAG 7, 21. mars: Anlegget ble kjørt med kun vann. Ny løsning på MQTT gjorde at sensorverdier ble logget og visualisert i Grafana. NGI (Norges Geotekniske Institutt) tok prøver av sedimentet og smeltevannet fra sedimenteringstanken for videre analyse.

DAG 8, 22. mars: Nedrigging av anlegget. Instrumentering ble koblet fra, og skapet ble tatt ut av brakken og fraktet til kontoret på VisionTech.

Bilder av hele anlegget ligger i Figur 23.

3.4 Databehandling og beregninger

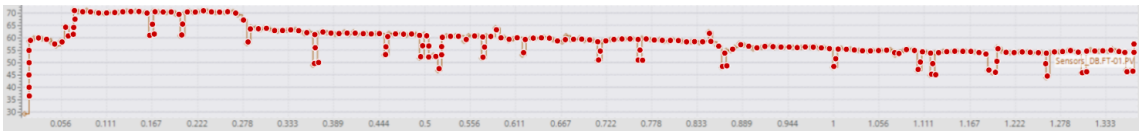
Ved utført testperiode er det loggført mye tidsseriedata. De innsamlede dataene benyttes for å beregne smeltekapasitet, samt lage en estimert empirisk modell. I dette kapittelet vises metodene som er benyttet for behandling av den loggførte tidsseriedataen. Fullstendig kode for de ulike beregningene ligger vedlagt i egen mappe.



Figur 19: Fremgangsmåte databehandling og beregninger

Figur 19 viser den overordnede fremgangsmåten for hvordan loggdataen er behandlet og benyttet.

Detektere datapunkter



Figur 20: Detektering av datapunkter vha maskinsyn [63]

Dataloggingen sviktet under kjøringen av parti 2 og parti 3 . Da måtte dataen hentes fra utklipp tatt fra trendlogging i PLSen ("Trace"). Figur 20 er et eksempel på hvordan datapunktene ble detektert ved hjelp av et nettbasert maskinsynsassistert verktøy [64] som senere ble eksportert til en CSV fil.

Filtrering og behandling av data

Ettersom detekteringen ikke var feilfri, måtte også dataene filtreres og behandles. Filtreringen ble gjort i Excel hvor datapunkter med tidsstempel mindre eller lik det forrige tidsstempel ble fjernet.

$$= HVIS(X4 >= X3; X4;) \tag{28}$$

I Likning 28 vises en enkel Excel-funksjon som returnerer 0 dersom nåværende tidsstempel er mindre eller lik det forrige. Deretter ble disse datapunktene fjernet manuelt.

Dataene ble så konvertert til en Pandas Dataramme i Python og normalisert rundt samme tidsstempel for hver av variablene til en verdi mellom 0 og 1.

For tidsseriedata for strømning i sjøen ble det laget en CSV fil med samme tidsstempel som det resterende datasettet, med konstant verdi 1000 rpm for parti 2 og 500 rpm for parti 3. Dette er fordi det ikke ble gjort målinger på strømninger i sjøen.

Beregning av smeltehastighet og plotting

Etter data er behandlet og plassert i CSV-filer, kan en ta i bruk verktøy som Pythonbiblioteket matplotlib eller MATLAB for å visualisere sensordata. Her kan en velge hvilket verktøy en bruker for å plotte. Prosjektgruppen har valgt å bruke MATLAB, siden det ga penere grafer enn matplotlib. Disse grafene presenteres i Kapittel 4.

Visualisering av sensordata kan gi indikasjon på når snøen ble dumpet i bassenget, samt hvor lang tid det tok før snøpartiet smeltet. Dette kan også sammenliknes med hva en observerer på testanlegget. Klokkeslettet når snøen dumpes i bassenget og når det er smeltet logges også manuelt for hvert parti.

For beregning av smeltehastighet;

$$Smeltehastighet\ [m^3/t] = \frac{Snø\ [m^3]}{Smeltetid\ [t]} \tag{29}$$

Lineær regresjonsmodell i MATLAB

Linære regresjonsmodeller kan gi en idé om hvilke smeltehastigheter en kan forvente på smelteanlegget, basert på omgivelsesvariablene; sjøtemperatur, lufttemperatur, strømning i rør og strømning i sjø. Kodesnutt 2 gir et eksempel på hvordan prosjektgruppen brukte MATLAB verktøyboksen "Statistics and Machine Learning Toolbox" for å tilpasse en regresjonsmodell fra innhentet data og predikere smeltehastigheter. De sentrale funksjonene tatt i bruk her er "fitlm" og "predict" [65]. "fitlm" er en MATLAB-funksjon som tar inn en tabell og returnerer en lineær regresjonsmodell. "predict" tar inn en lineær modell, samt en tabell med nye datapunkter, og returnerer de forutsatte responsverdiene.

```
data = table(MeanAirTemp, MeanSeaTemp, MeanFlowRate, SjostromRPM,
↪  MeltingRate, ...
        'VariableNames', {'AirTemp', 'SeaTemp', 'FlowRate', 'BoatRPM',
        ↪  'MeltRate'});

lm = fitlm(data, 'MeltRate ~ AirTemp + SeaTemp + FlowRate + BoatRPM');

newData = table(6, 5, 50, 700, 'VariableNames', {'AirTemp', 'SeaTemp',
↪  'FlowRate', 'BoatRPM'});

predictedMeltingRate = predict(lm, newData);
```

Kodesnutt 2: «fitlm» og «predict» funksjon fra MATLAB

Lineær regresjonsmodell i Python

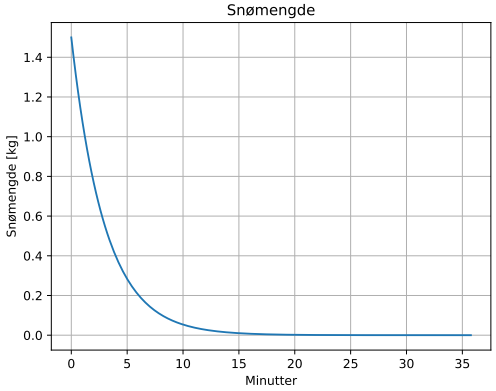
For å vite hvilken innvirkning de ulike variablene hadde på systemet ble datasettet benyttet til å trene en maskinlæringsmodell ved hjelp av Pythonbiblioteket "sklearn" (også kjent som "Scikit-learn").

Tidspunktene snø blir dumpet i konteineren på, er tilfeldige, og fører til store endringer i datasettet. Dette er uforutsette endringer for maskinlæringsmodellen, og den vil derfor prøve å finne en

sammenheng mellom endringene og de andre variablene. Denne sammenhengen eksisterer ikke, så det er derfor hensiktsmessig å koble endringene opp mot snømengde i systemet. For å gjøre dette ble det simulert en impulsrespons til en første ordens prosess med tidskonstant τ på 3 minutter, og proporsjonalitetskonstant k på 1.5 $[kg/m^3]$. Dataen ble logget med samme intervall som resten av datasettene, som vist i Kodesnutt 3.

```
time = np.linspace(0, 35.76352664, 179)

def impulse_response(t, k, tau):
    return k * np.exp(-1/tau * t)

values = impulse_response(time, 1.5, 3)
```

Kodesnutt 3: Simulert første ordens impulsrespons for snømengde



Figur 21: Simulert data for snømengde

Impulsrespons plottes som vist i Figur 21, og etter omtrent 10 minutter, eller 3-4 tidskonstanter er verdien tilnærmet 0. Responsen er basert på både observasjonen fysisk under testing og fra målinger av returtemperaturen under parti 2. Dataene lagres og benyttes videre som en del av datasettet.

```
START_TIME = 0
MAX_TIME = 25

X = data[['luft2', 'sjø2', 'flow2', 'flow_sjø2', 'snømengde', 'luft3',
↪  'sjø3', 'flow3', 'flow_sjø3']]

Y = data['retur2'] + data['retur3']

train_model(X, Y, START_TIME, MAX_TIME)
```

Kodesnutt 4: Lineær regresjonsanalyse av tidsseriedata

I Kodesnutt 4 vises en overordnet metode for trening av maskinlæringsmodell basert på tidsseriedataen fra parti 2 og parti 3. Koden er skrevet i Python hvor blant annet metodene "LinearRegression()" og "train_test_split()" er brukt [66] [67].

ARIMA modelltilpasning

En annen metode som egner seg til identifisering av tidsseriedata, og som gir bedre svar på vekting av de ulike tilstandene i systemet er ARIMA-modell fra Python biblioteket "pmdarima". Denne modellen ble benyttet på datasettet på tilsvarende måte som maskinlæringsmodellen for å gi ytterligere svar på systemets dynamikk.

```
import pmdarima as pm
auto_arima = pm.auto_arima(Y, X=X, seasonal=False, stepwise=True)
```

Kodesnutt 5: Automatisk ARIMA modellidentifikasjson

I Kodesnutt 5 benyttes det samme datasettet i en funksjon for automatisk ARIMA modelltilpasning, "auto_arima()". Funksjonen tar inn datasettet og andre eventuelle spesifikasjoner og gjør et rutenettsøk for ulike verdier av parametrene p, d og q. Til slutt velges modellen som minimerer for relevante ytelsesmetrikker.

4 Resultater

Kapittelet presenterer endelige resultater fra prosjektet dokumentert med bilder og kodesnutter. Først presenteres anlegget med tilhørende bilder fra testperioden og av automasjonstavlen. Deretter programvare for PLS, alarmsystemer, HMI og IT-systemer, samt målinger og beregninger fra testperioden.

4.1 Anlegg

Anlegget ble rigget opp på Fagervika kai, og består av en arbeidsbrakke, smeltebinge, veksler, dieselgenerator, båt og kranbil. Prosjektgruppen utførte målinger på tre partier med snø hentet fra et snødeponi på Tiller i Trondheim, og gjennomførte testperioden sammen med RebelGarden, VisionTech og Betongbygg Trøndelag. AF Gruppen var også involvert med utlån av arbeidsbrakke, konteiner, slampumpe og diverse utstyr.

Prosjektgruppen plasserte automasjonstavlen (Figur 24) i en arbeidsbrakke stående sentralt på anlegget. Fra tavlen var det mulig å styre og få informasjon om systemet på operatørpanelet. En åpen konteiner (Figur 23a) ble plassert på land. En hjullaster tømte snø i konteineren ved start av nytt parti. Deretter ble slush pumpet gjennom en slange, gjennom veksleren, opp til en ny slange og tilbake i konteineren. Veksleren var senket ned i sjøen ved hjelp av en kranbil. Figur 23b viser en båt som ble fortøyd til land og som ble brukt for å skape en strømning i vannet rundt veksleren. Dette ble gjort for å unngå at kaldt vann la seg rundt rørene. Anlegget brukte et dieselaggregat for strømforsyning, ettersom el-anlegget på kaien ikke var dimensjonert for den høye startstrømmen på pumpen.

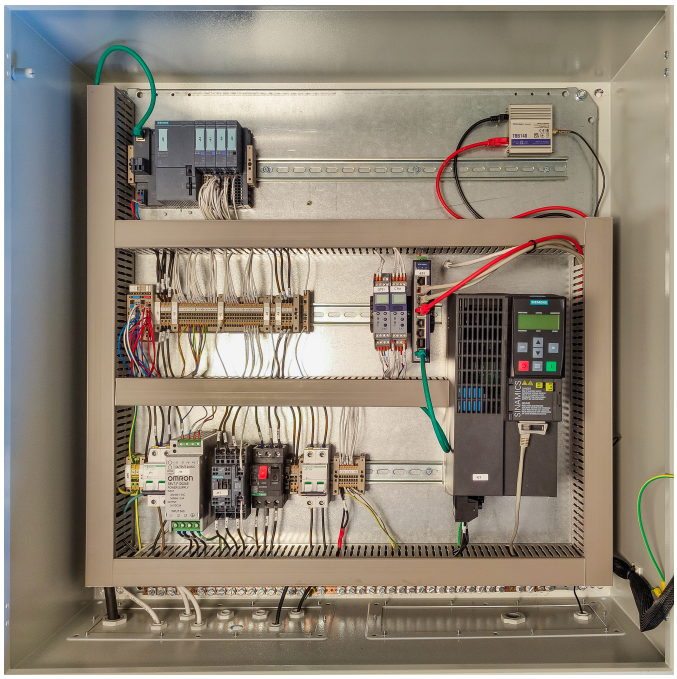
Figur 22: Veksler som senkes i sjøen under drift

(a) Nytt parti med snø i bassenget *(b) Veksler og båt*

Figur 23: Anlegget

Automasjonstavle

Tavlen i Figur 24 er bygget av prosjektgruppen med veiledning fra VisionTech. På bildet vises PLSen sammen med I/O-modulene, frekvensomformeren, transmittere for konduktivitet- og pH-sensorer, motorvern, kontaktor, transformator, 400 V sikring og 230 V sikring, ethernet-svitsj, Teltonika IoT-modul, og diverse rekkeklemmer. Dette er montert på DIN-skinner i et 800 mm x 800 mm x 240 mm skap. I døren på skapet er det skåret ut plass til et 7-tommers HMI-panel. På undersiden av skapet er det boret ut hull til strømtilførsel, uttak fra frekvensomformer, sensorer, ethernetkabel og en antenne fra Teltonika.

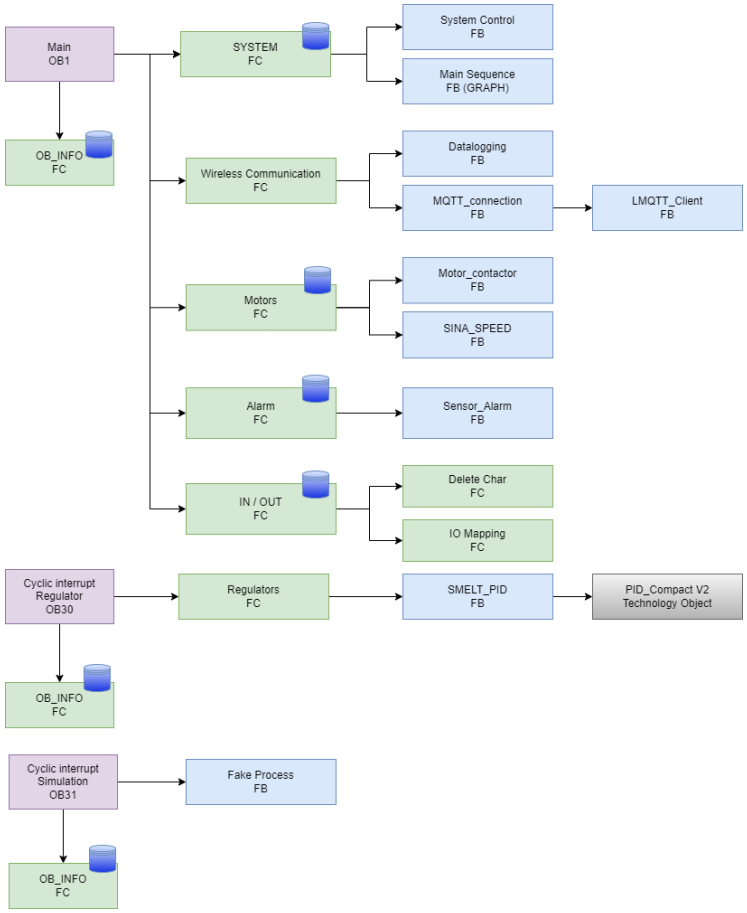


Figur 24: Automasjonstavle innvendig

Figur 25: Automasjonstavle utvendig

4.2 Programvare: PLS

Dette underkapittelet tar for seg resultater fra programvare i PLS, og er videre inndelt i; motorstyring, reguleringsalgoritmer, sekvensprogram og dataformatering.

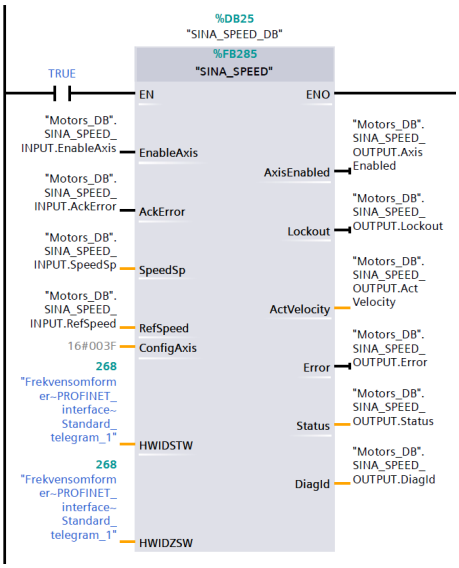


Figur 26: Programstruktur i TIA Portal

Figur 26 viser overordnet programstruktur for PLS programmet, og følger prinsippene presentert i Kapittel 3.2.1 for å oppnå en oversiktlig og skalerbar struktur.

Motorstyring

For å styre frekvensomformeren gjennom programvaren er det brukt "SINA_SPEED"; et ferdig utviklet bibliotek fra Siemens. Figur 27 viser innganger og utganger av funksjonsblokken, som kommuniserer med frekvensomformeren over "Standard Telegram 1".



Figur 27: SINA_SPEED motorblokk

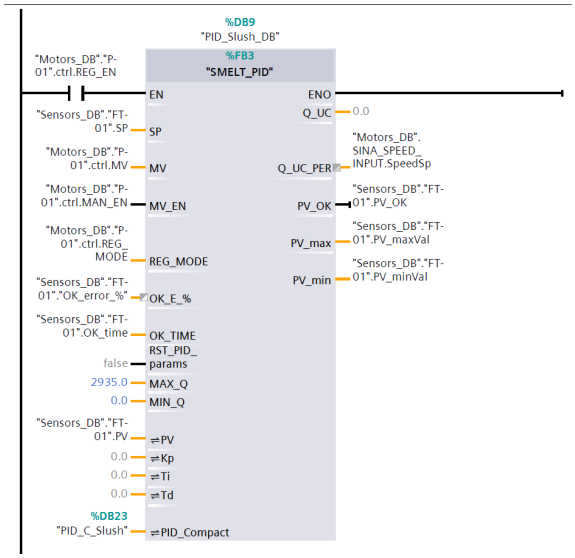
Venstre side av funksjonsblokken representerer data som sendes til frekvensomformeren, og høyre side representerer data motatt. Dataene som sendes er; pumpens nominelle turtall, ønsket turtall, alarmkvittering og start/stopp av drivverket. Dataene som mottas er; drivverk aktivert, sikkerhetsstopp, målt turtall, avvik, status og diagnostikk ID.

Ønsket turtall hentes fra reguleringsblokken som bestemmer turtall basert på målingene fra strømning i slushen. Dette resultatet beskrives nærmere under "Reguleringsalgoritmer" i Kapittel 4.2.

Dataene som mottas er nyttig for bruk av feilsøking, og brukes også i alarmsystemet til PLSen.

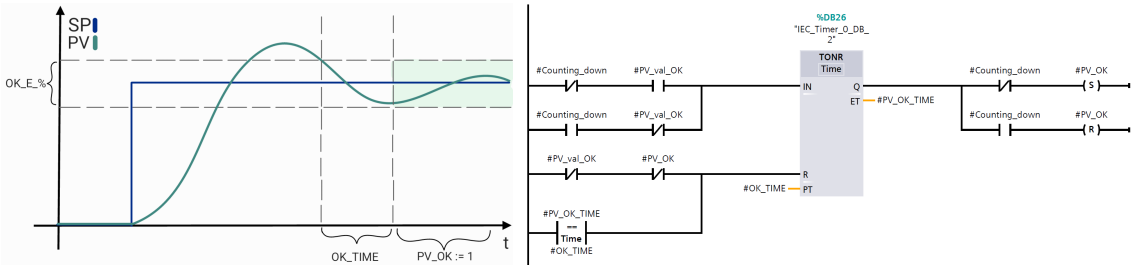
Reguleringsalgoritmer

For å kunne regulere flere prosesser ble det laget en generell reguleringsblokk "SMELT_PID" av typen "Parameter Instans" som benytter en instans av teknologiobjektet PID_Compact som inn/ut variabel. På denne måten er det enkelt å gjenbruke funksjonaliteten i funksjonsblokken for andre prosesser som skal reguleres.



Figur 28: Utdrag fra reguleringsblokk til slushpumpe

Inngangsparameterne "#OK_E_%" og "#OK_TIME" definerer når reguleringsavviket ansees å være innenfor ønskede verdier. Dersom man for eksempel setter "#OK_E_%" lik 10, og "#OK_TIME" til #T10s så vil utgangsparameteren "#PV_OK" gå høy dersom prosessverdien har vært innenfor 10% av referansen i minst 10 sekunder. Figur 29a illustrerer dette, og Figur 29b viser deler av logikken som brukes. Her gjenbrukes samme tidtaker for både å telle opp før "#PV_OK" er høy, samt for å telle ned etter den har blitt satt høy.



(a) Illustrasjon: Reguleringsavvik *(b) Logikk: Reguleringsavvik*

Figur 29: Virkemåte til reguleringsavvik

Simulert diskret prosess

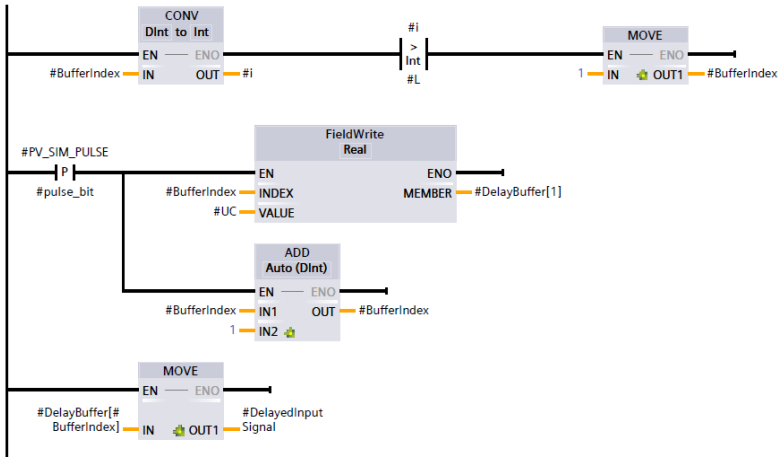
For å teste reguleringssystemet har det blitt laget en gjenbrukbar simulert første ordens prosess-modell med tidsforsinkelse (FOPTF). Modellen kjøres parallelt med resten av PLS-programmet i en egen Organisasjonsblokk av typen "Cyclic Interrupt". Denne kjøres syklisk for å sikre pålitelig oppførsel.

```
1        #PV := TRUNC(1000*(#PV + (#k/#tau)*(#DelayedInputSignal - #PV)*#Ts))/1000;
```

Kodesnutt 6: Simulert første ordens prosess med tidsforsinkelse

Hvor:

TRUNC()	Innebygd funksjon	Returnerer avrundede heltallsverdier
#PV	InOut	Prosessverdi til simulert prosess
#k	Input	Proporsjonalkonstant
#tau	Input	Tidskonstant
#Ts	Input	Tastetid i aktuell OB
#DelayedInputSignal	Temp	Forsinket pådrag



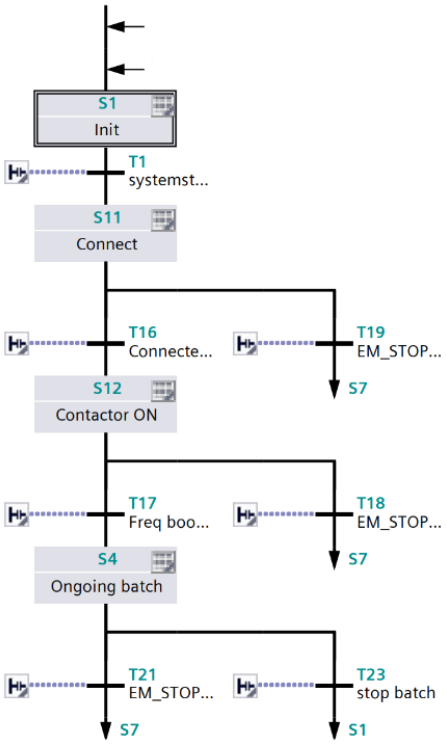
Figur 30: Logikk for forsinkelsesbuffer

I Kodesnutt 6 skjer selve utregningen av den simulerte prosessen i form av en diskret FOPTF. Tidsforsinkelsen er simulert ut ifra logikken i Figur 30 hvor verdier fra pådraget levert av regulatoren lagres i en forsinkelsesbuffer med variabel størrelse $\#L$. Ved å endre på parametrene $\#k$, $\#tau$ og $\#L$ kan man tilpasse responsen til prosessen etter ønsket dynamikk. Tastetiden $\#Ts$ hentes direkte fra syklustiden til organisasjonsblokken som funksjonen kjører i.

Sekvensprogram

TIA Portal har en innebygd funksjonsblokk for programmering av sekvensiell programflyt. Denne bruker Siemens' egne programmeringspråk for SFC som heter GRAPH. Figur 31 viser sekvensdiagrammet for prosjektet. Den gjør det enklere å få oversikt over hendelseforløpet til systemet.

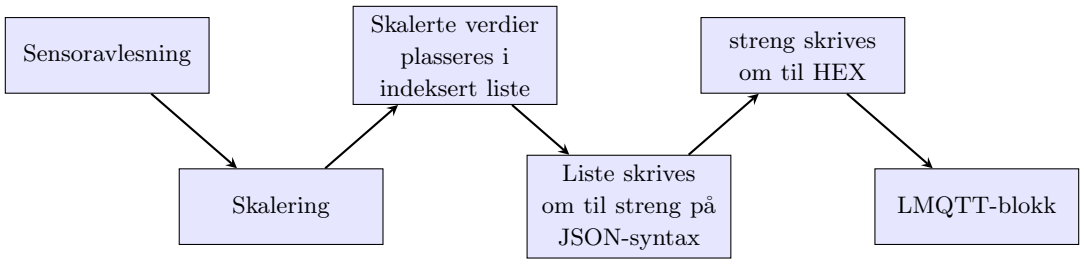
Sekvensdiagrammet starter med å initialisere alle verdier. Når det logges inn på HMIen vil programmet gå videre til neste steg, "Connect". Her kobles PLSen til server for datalogging. For at kontaktoren skal slå inn, må nødvendige verdier for kjøring settes. Dette gjelder referanseverdi for "slush strømning" og "båtmotor". Når oppkoblingen er gjennomført vil kontaktoren slås inn og vente på at frekvensomformeren starter opp. Frekvensomformeren setter bit 2 i "Status ord 1" til høy når den klar for styring. Det er hele tiden mulig å trykke på "NØDSTOPP" i HMIen for å gå tilbake til steget "Init", der kontaktoren slår ut.



Figur 31: Overordnet sekvensdiagram. Programmert i SFC, GRAPH

Dataformatering

Dette underkapittelet tar for seg alle programvareleddene fra sensoravlesning til publisering av data til MQTT-megleren. For å kunne gjøre dette må dataen formateres gjennom ulike steg. Programvaren er utviklet i TIA Portal, og tar i bruk egen kode, samt et bibliotek for publisering over MQTT. Figur 32 gir en oversikt over leddene involvert.



Figur 32: Flytskjema for datalogging i PLS

Figur 33 viser PLS-adressene sensorverdiene hentes fra. RTD sensorene gir verdier $10x$ den reelle verdien, fordi datatypen er *int* og håndterer derfor ikke desimaler. Verdien fra FT_01 kommer som en *int* mellom $0 - 27648$ der signalverdiene går fra $4\ mA - 20\ mA$.

Sensors [6]

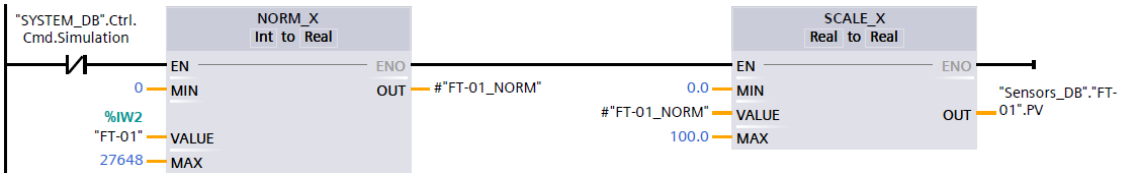
PLC tags

	Name	Data type	Address	Comment
	TT-01_Retur	Int	%IW8	Sensor - returtemp
	TT-02_Ute	Int	%IW10	Sensor - utetemp
	TT-03_Sjø	Int	%IW12	Sensor - sjøtemp
	QT-01_pH	Int	%IW16	Sensor - pH
	CT-01_C	Int	%IW18	Sensor - konduktivitet i vann
	FT-01	Int	%IW2	Sensor - Flow i rør

Figur 33: PLS-tags for sensoravlesing

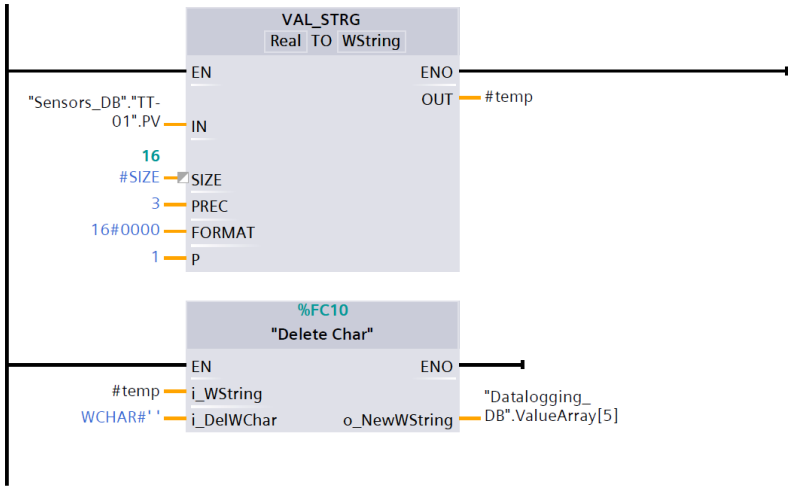
Kodesnutt 7 Viser hvordan sensorverdiene regnes om til prosessverdier. RTD-sensorene skaleres med $\frac{1}{10}$ og FT_01 regnes over til m^3/t.

```
1    "Sensors_DB"."TT-01".PV := INT_TO_REAL("TT-01_Retur")/ 10;
2    "Sensors_DB"."TT-02".PV := INT_TO_REAL("TT-02_Ute") / 10;
3    "Sensors_DB"."TT-03".PV := INT_TO_REAL("TT-03_Sjo") / 10;
4    "Sensors_DB"."QT-01".PV := INT_TO_REAL("QT-01_pH") / 10;
5    "Sensors_DB"."CT-01".PV := INT_TO_REAL("CT-01_C") / 10;
```



Kodesnutt 7: Skalering av sensoravlesinger

Figur 34 viser et eksempel på hvordan en skalert sensorverdi blir plassert i en indeksert liste. Funksjonen FC10, "Delete Char", er en egenlaget funksjon som fjerner et valgt tegn fra en "WString". Her settes størrelsen på sensorverdien til 16 bit med fast presisjon på 3 desimaler, og deretter sløyfes alle tomme tegn (' '). På denne måten kan alle verdier opp mot 16 bit håndteres likt.



Figur 34: Plasserer sensorverdier i en liste

Kodesnutt 8 sammenfatter alle verdier fra den indekserte listen til en meldingspakke formatert som en JSON-streng. Et eksempel på hvordan denne JSON-strengen ser ut vises i Kodesnutt 12 i Kapittel 4.5.

```
1        #message_WString := STRING_TO_WSTRING('{');
2
3        FOR #j := 1 TO #NUMB_OF_MSGS DO
4            // Start med nokkelen, som alltid er en streng
5            #message_WString := CONCAT_WSTRING(IN1 := #message_WString, IN2 :=
                 ↪ STRING_TO_WSTRING('"'), IN3 := #KeyArray[#j], IN4 :=
                 ↪ STRING_TO_WSTRING('": '));
6            // Siden alle verdier skal vaere tall, trenger vi ikke anforselstegn
                 ↪ rundt verdiene
7            #message_WString := CONCAT_WSTRING(IN1 := #message_WString, IN2 := #
                 ↪ ValueArray[#j]);
8
9            IF #j < #NUMB_OF_MSGS THEN
10               // Legger til komma og mellomrom for alle unntatt det siste
                     ↪ elementet
11               #message_WString := CONCAT_WSTRING(IN1 := #message_WString, IN2 :=
                     ↪ STRING_TO_WSTRING(', '));
12           END_IF;
13       END_FOR;
14
15       #message_WString := CONCAT_WSTRING(IN1 := #message_WString, IN2 :=
             ↪ STRING_TO_WSTRING('}'));
```

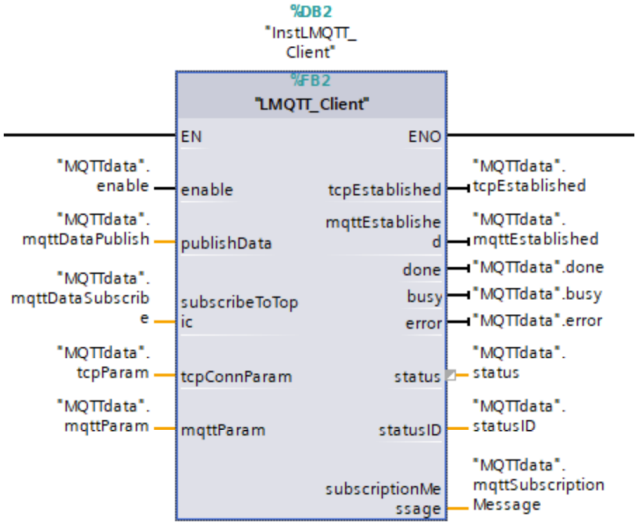
Kodesnutt 8: Gjør om liste til en streng skrevet på JSON-syntax

Kodesnutt 9 viser hvordan hvert tegn i JSON-strengen konverteres til hexadesimal og plasseres i en indeksert liste. Dette gjøres fordi LMQTT- funksjonsblokken kun tar imot meldingspakker på formatet hexadesimal. Denne listen er meldingspakken som publiseres på MQTT-klienten

```
#message_string_length := LEN(#message_WString);

"MQTT_connection_DB".pubMessageCnt := INT_TO_UDINT(#message_string_length);

IF "SYSTEM_DB"."04 Wireless Communication".MQTT_Con_Status = 1 THEN
    FOR #i := 1 TO #message_string_length DO
        "MQTT_connection_DB".message[#i - 1] :=
        WCHAR_TO_BYTE(WSTRING_TO_WCHAR(#message_WString[#i]));
    END_FOR;
END_IF;
```

Kodesnutt 9: Konverterer JSON-streng til hexadecimaler.

Funksjonsblokken i Figur 35 publiserer meldingspakken til MQTT-megleren på Teltonika enheten.



Figur 35: LMQTT-bibliotek som publiserer meldingspakkene.

4.3 Alarmsystem

Strukturen til alarmsystemet tar i bruk funksjonaliteten "HMI-alarms", sammen med kodesnutter i PLS-programmet. For å skille typer alarmer og alvorlighetsgrad, sorteres alarmene i klasser. Mindre alvorlige feil som ikke påvirker selve driften av anlegget er i klassen "Warnings", og vises kun i alarmlisten på panelet dersom de er aktive. Dette gjelder blant annet for lav strømning i rør, og dersom OB80 eller de andre OBene, fra Kapittel 2.4, er aktive. Alvorlige feil er i klassen "Error", og må kvitteres for å fjernes fra alarmlisten. Det samme gjelder sensoralarmer, som har den egendefinerte alarmklassen "Sensor error". Forhåndsdefinerte alarmer for system- og programfeil vises også i alarmlisten så lenge de er aktiv. Disse trenger ikke å kvitteres. Flere alarmsituasjoner har samme trigger tag, men trigges av ulik bit, definert i en DB. I Figur 36 vises et utklipp av hvordan de diskrete alarmene er konfigurert, både med egen og felles trigger tag.

Discrete alarms

ID	Name	Alarm text	Alarm class	Trigger tag	Trigger bit	Trigger address
2	SINA_error_1	Drive fault active	Warnings	Alarm_DB_Frekvensomformer_Trigger_Tag	1	Alarm_DB.Frekvensomformer.Trigger_Tag.x1
3	SINA_error_2	On-inhibit active	Warnings	Alarm_DB_Frekvensomformer_Trigger_Tag	2	Alarm_DB.Frekvensomformer.Trigger_Tag.x2
4	SINA_error_3	On-inhibit active	Warnings	Alarm_DB_Frekvensomformer_Trigger_Tag	3	Alarm_DB.Frekvensomformer.Trigger_Tag.x3
5	SINA_error_4	DPWR_DAT error	Warnings	Alarm_DB_Frekvensomformer_Trigger_Tag	4	Alarm_DB.Frekvensomformer.Trigger_Tag.x4
6	MQTT N/C	MQTT not connected	Warnings	MQTT_connection_DB_Con_status	1	MQTT_connection_DB.Con_status.x1
7	MQTT Error	MQTT error	Errors	MQTT_connection_DB_Con_status	2	MQTT_connection_DB.Con_status.x2
8	OB 80 error	Time error interrupt	Warnings	Alarm_DB_OB_Errors_OB 80	1	Alarm_DB.OB_Errors."OB 80".x1
9	OB 82 error	Diagnostics error	Warnings	Alarm_DB_OB_Errors_OB 82	1	Alarm_DB.OB_Errors."OB 82".x1
10	OB 122 error	IO Access error	Warnings	Alarm_DB_OB_Errors_OB 122	1	Alarm_DB.OB_Errors."OB 122".x1
11	OB 121 error	Programming error	Warnings	Alarm_DB_OB_Errors_OB 121	1	Alarm_DB.OB_Errors."OB 121".x1
12	OB 83 error	Pull or plug of modules	Warnings	Alarm_DB_OB_Errors_OB 83	1	Alarm_DB.OB_Errors."OB 83".x1
13	OB 86 error	Rack or station failure	Warnings	Alarm_DB_OB_Errors_OB 86	1	Alarm_DB.OB_Errors."OB 86".x1
14	EMERGENCY STOP	NØDSTOPP: Ack this alarm to continue	Errors	Alarm_DB_EMERGENCY_STOP_Trigger_tag	1	Alarm_DB.EMERGENCY_STOP.Trigger_tag.x1

Figur 36: Oppsettet i HMI-alarms. Deler av de diskrete alarmer

Det ble laget en funksjonsblokk for å detektere feil med sensorene. Funksjonsblokken "Sensor_Alarm [FB7]", sjekker sensorverdien, og trigger en tag ved avvik. På den måten kan sensoravvik ha én diskret alarm pr. sensor.

For å samle kode relatert til alarmsystemet, ble det laget en funksjon "Alarm [FC2]". Her detekteres avvik og lagres i en trigger tag, som deretter konfigureres som alarmer i *HMI-alarms*. Det ble laget en "Alarm_DB" for organisering av disse taggene. Funksjonen inneholder blant annet kode for å detektere avvik med MQTT-kommunikasjon og feil med frekvensomformeren. Andre avvikssituasjoner har lignende oppsett som vist i Kodesnutt 10.

```
1  IF "MQTT_connection_DB".Output.status <> 16#7004 THEN
2      "Alarm_DB".Communication.Trigger_Tag := 1;
3      #found := TRUE;
4   END_IF;
5
6   IF NOT #found THEN
7      "Alarm_DB".Communication.Trigger_Tag := 0;
8  END_IF;
```

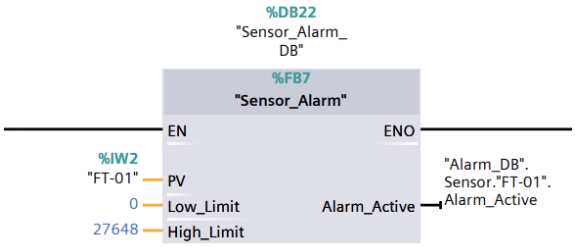
Kodesnutt 10: Sjekker om MQTT er tilkoblet. Setter trigger tag aktiv hvis ikke.

> *I Kodesnutt 10 sjekkes det for om variabelen "MQTT_connection_DB".Output.status er ulik hextallet* 16#7004. *Dette tallet representerer statuskoden for at klientens tilkobling til serveren er suksessfull.*

Funksjonsblokken "Sensor_Alarm [FB7]" brukes også inne i funksjonen "Alarm [FC2]", og benytter datablokken for alarmer. Inngangsverdiene til blokken er sensorens prosessverdi, øvre og nedre grense. Utgangen er trigger-taggen for alarmen. Samtlige sensorer bruker funksjonsblokken, og øvre og nedre grense for prosessverdi er tatt fra sensorens datablad. Kodesnutt 11 viser innholdet i "Sensor_Alarm [FB7]" , og Figur 37 viser et eksempel på hvordan blokken er ut og brukes.

```
IF #PV < #Low_Limit OR #PV > #High_Limit THEN
    #Alarm_Active := 1;
END_IF;
```

Kodesnutt 11: Innholdet i Sensor_Alarm[FB7]



Figur 37: Bruken av Sensor_Alarm [FB7] for sensor FT-01

4.4 Programvare: HMI

Prosjektgruppen har laget HMI utifra ønsker fra oppdragsgiver. Dette innebærer spesifikasjoner for et sekvensielt forløp, hva systemsiden skulle inneholde og noe generell funksjonalitet som alarm og trend. Gruppen har utenom dette kunne tatt seg mye friheter i utforming av HMIen. Standarder for farger og design blir brukt for å møte kravene for "høy ytelse", forklart i Kapittel 2.5. "Norm for symboler i driftskontroll-systemer for VA-sektoren" fra NORVAR [68] blir også brukt.

Operatørpanelet har to sikkerhetsnivåer; full tilgang og begrenset tilgang. Under disse er det lagt inn ulike brukere med de ulike tilgangene. Det er mulig å legge til et ubegrenset antall med brukere. Nye brukere må legges inn i programmet i TIA Portal og lastes opp til HMI-panelet. De ulike sikkerhetsnivåene vil ha ulik tilgang for navigering rundt i operatørpanelet.

Innenfor sikkerhetsgruppen "Full tilgang" vil man kunne operere og monitorere hele systemet på brukerpanelet. Denne operatøren vil ha en spesiell tilgang til en kontrollside der det ligger en del informasjon og innstillinger som ikke er nødvendig i normal drift. Kontrollsiden brukes hovedsaklig til å se at alt fungerer som det skal og feilsøking. Dette innebærer informasjon om kommunikasjon, kjøringstid og frekvensomformer.

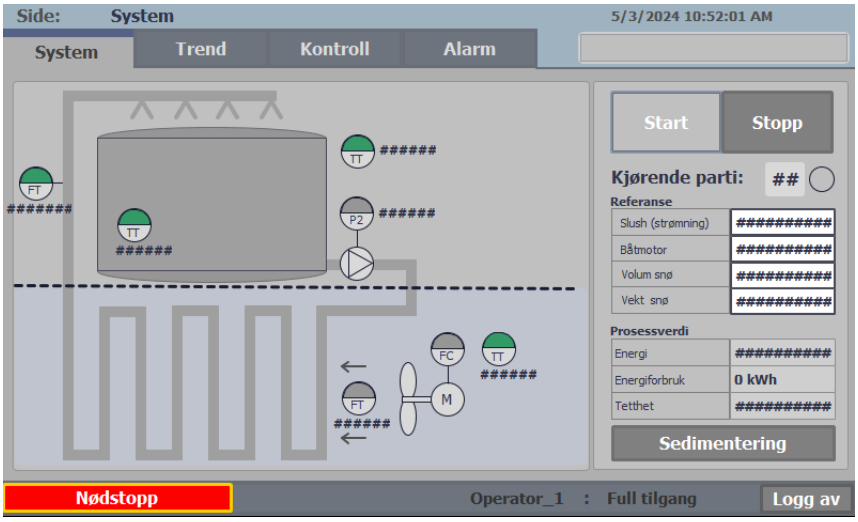
Systemside

Figur 38 viser et bilde av systemsiden. Alle verdier står som "####", siden bildene er tatt under simulering i TIA Portal og ikke er koblet opp til selve PLSen. Systemsiden blir vist etter å ha logget inn på HMI-panelet. Her vises en oversikt over hele systemet, inspirert av skissen i Figur 23. Systemet har riktige indikatorer for ulike komponenter ifølge krav for høy ytelse, og viser alle nødvendige verdier fra sensorer. Feltene som er hvite er mulig å skrive til, mens de grå viser verdier som blir hentet fra sensorer eller programmet. Referanseverdier fylles inn manuelt. Det er mulig å operere anlegget fra systemsiden og alle operatørene har derfor tilgang til denne siden.

Skrifttypen som blir brukt gjennom hele HMIen heter "Tahoma". Dette er en Sans-Serif skrifttype Kapittel 2.5. I TIA Portal er det mulig å velge mellom to ulike skrifttyper "Tahoma" og "Courier New", i tillegg til at disse kan være i normal tykkelse, fet skrift eller kursiv. Skrifttypen som velges skal være av typen Sans-serif. Det er en "[...] samlebetegnelse på skrifttyper som mangler seriffer, det vil si mangler tverrstreker eller 'føtter' på enden av hovedstrekene" [69]. Dette anses som skrifttyper som er enkle å lese og fungerer derfor bra i et brukergrensesnitt hvor man skal kunne navigere seg rundt raskt.

Gjennom hele HMIen vil det alltid være en Meny-linje der en kan navigere mellom de ulike sidene. Her er det tydelig hvilken side som er aktiv. Dette er også synlig øverst til venstre, der det vil

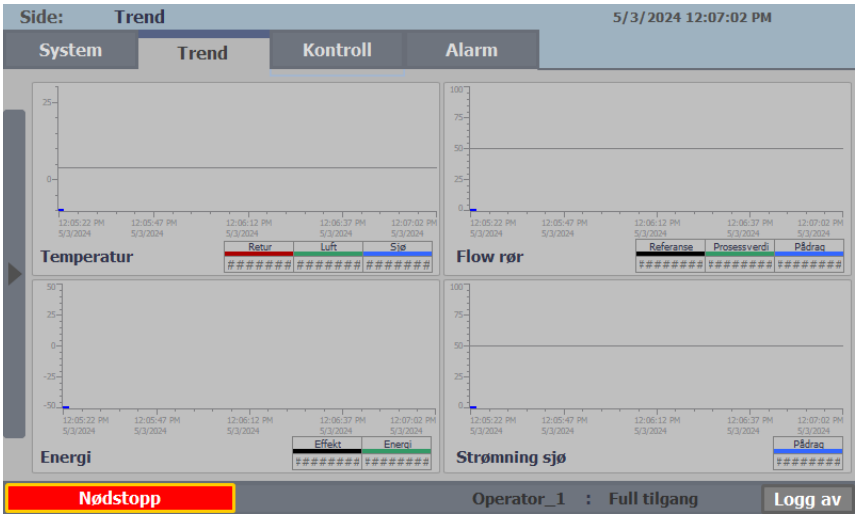
stå hvilken side som er aktiv. Ved siden av menylinjen er det et lite alarmvindu som er synlig hele tiden. Nederst på HMIen er det alltid en statuslinje som viser hvilken operatør som er aktiv med tilhørende tilgangsnivå. På denne linjen finner man også en nødstopp-knapp. Når denne blir trykket slår kontaktoren ut slik at frekvensomformeren og pumpen stopper. Det blir da gitt en varsling, i form av et sprett-opp-vindu, om at det er gjennomført en nødstopp. Når denne alarmen aktiveres må en operatør kvittere den ut før systemet kan startes opp igjen.



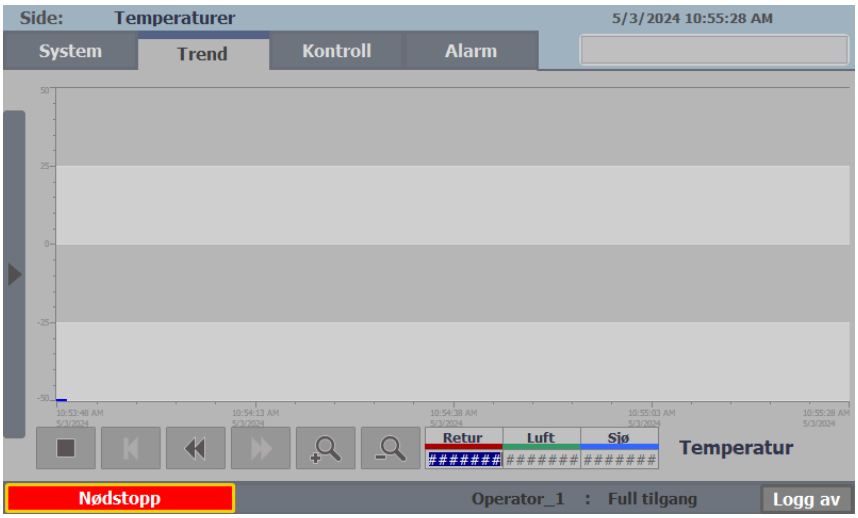
Figur 38: HMI systemside

Trendside

På trendsiden får en opp fire trendvinduer med aktuelle verdier. Disse viser de ulike temperaturene vi henter; strømning i rør, energiforbruket og strømning i sjø. Hver av disse vil være mulig å trykke seg inn på slik at en får se en forstørret versjon av det aktuelle trendvinduet, som vist i Figur 40. Til venstre i bildet er det lagt inn et innskyvningsvindu som vil vise de satte parameterne til systemet (mye av det samme som også vises på systemsiden) hvis den blir dratt ut.



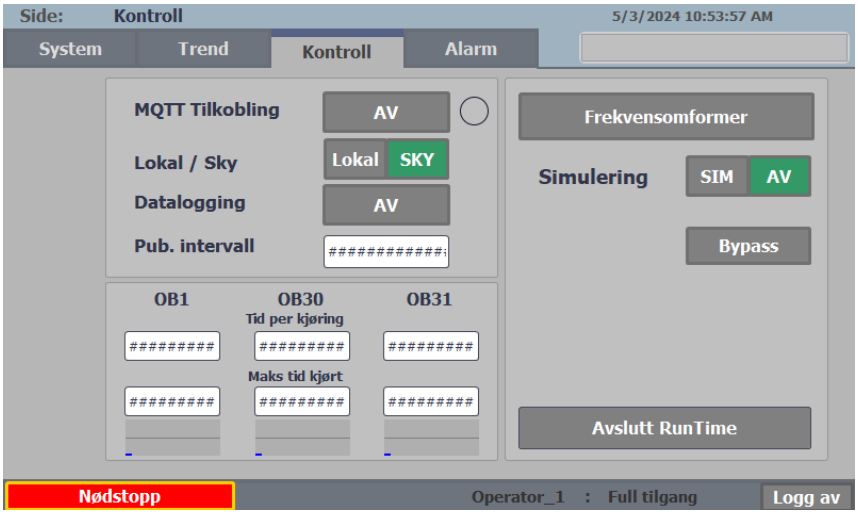
Figur 39: HMI trendside



Figur 40: HMI trendside for temperaturer.

Kontrollside

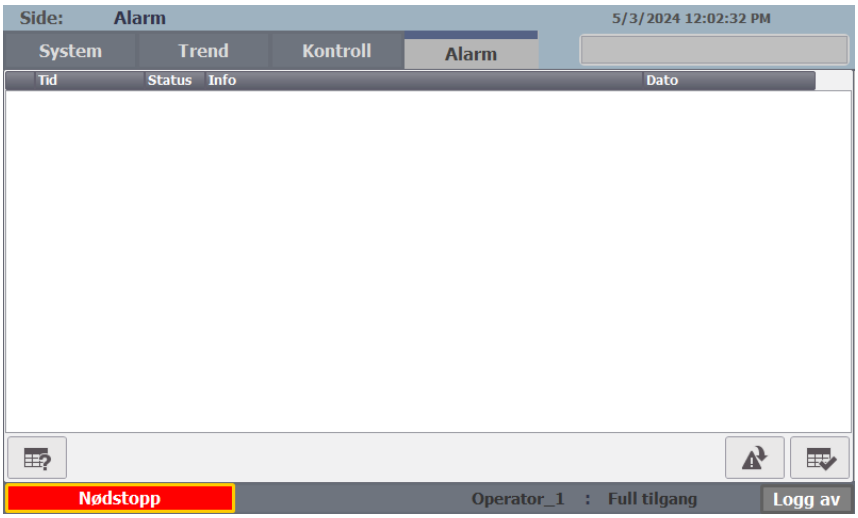
Kontrollsiden er kun tilgjengelig for operatører med avansert tilgangsnivå. Her vil en få en dypere innsikt i systemet. Navigering i logging, visualisering av tastetid og informasjon om frekvensomformeren vises for å kontrollsjekke parametere og for å kunne utføre noe feilsøking uten å måtte bruke en tilkoblet pc. Denne siden vil fortsatt få flere endringer fordi den brukes under utvikling og testing av systemet.



Figur 41: HMI kontrollside

Alarmside

Alarmsiden viser et alarmvindu med nødvendige alarmer. Alarmene kan deles inn i to ulike grupper; alarmer (error) og advarsler (warnings). Advarsler er kun synlige i alarmlisten når de er aktive og trenger ikke kvitteres. Alarmer er mer kritiske og vises dermed i alarmvinduet helt til de er inaktive og kvittert. Om de har en annen status enn inaktiv og kvittert samtidig, vil de ha en farge tilhørende deres status som er beskrevet i Tabell 9. Når en alarm eller advarsel er markert i alarmvinduet kan operatøren trykke på knappen nederst til venstre for å få opp mer informasjon om alarmen. Knappen nederst til høyre kvitterer alarmen.



Figur 42: HMI alarmside

Type alarm	Farge
NØDSTOPP, aktiv	Rød
Error, aktiv	Rød
Error, aktiv og kvittert	Lys rød
Error, inaktiv og ikke kvittert	Hvit
Sensor error, aktiv	Rød
Sensor error, aktiv og kvittert	Lys rød
Senror error, inaktiv	Hvit
Warning, aktiv	Oransj
Øvrig alarm	Lys blå

Tabell 9: Oversikt over alarmfarge

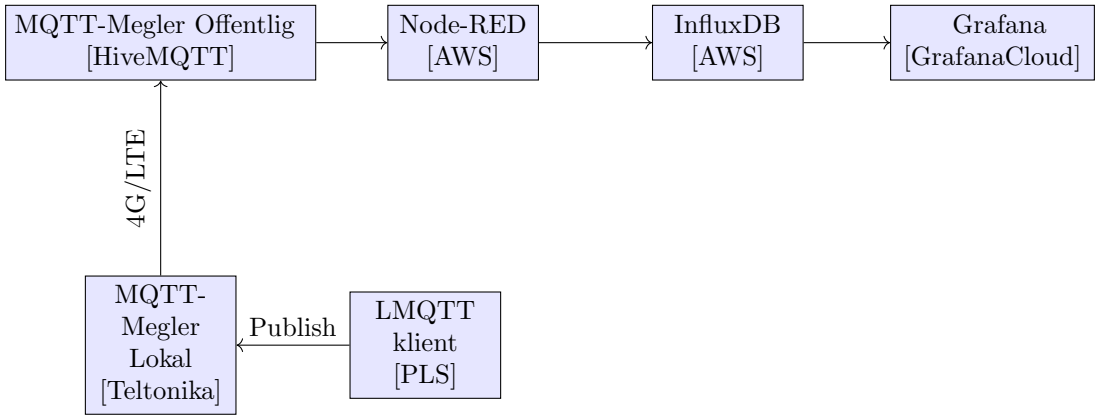
4.5 Programvare: IT-løsninger

IT-systemet som sikrer kommunikasjon og datalogging består av en rekke programvareløsninger listet i Tabell 10. Deler av løsningen kjøres lokalt på maskinvare og resten i betalte skytjenester. I prosjektgruppens tilfelle ble Amazon Web Services tatt i bruk for å drifte InfluxDB og Node-RED, og Grafana Cloud for å drifte Grafana. Overgangen fra lokalt nett til sky skjer ved hjelp av en MQTT-bro installert på Teltonika IoT-modulen. Hele kommunikasjonsrekken foregår som enveiskommunikasjon, men mulighetene for toveiskommunikasjon finnes, da MQTT er en to-veis kommunikasjonsprotokoll.

LMQTT-klient	Lokal på PLS
MQTT-megler	Lokal på Teltonika
MQTT-megler	Offentlig på HiveMQTT
Node-RED	I skyen på AWS
InfluxDB	I skyen på AWS
Grafana	I skyen på GrafanaCloud

Tabell 10: Programvareløsninger som brukes i kommunikasjon og datalogging

I Figur 43 illustreres løsningen på kommunikasjonsarkitekturen. Dette er en av mange mulige løsninger prosjektgruppen var inne på. Disse diskuteres nærmere i Kapittel 5.2.



Figur 43: Kommunikasjon, detaljert arkitektur

Sammenfatte data i publiserbare meldingspakker

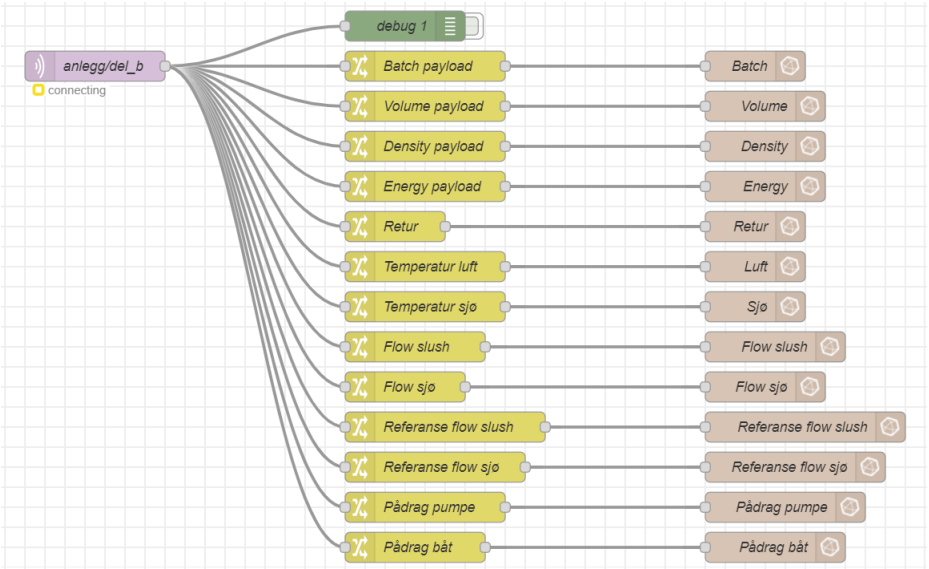
Sensordata og andre relevante data skrives inn i en liste som senere samles til en streng i PLS programmet og sendes over MQTT-protokollen ved hjelp av en LMQTT-klient, nærmere forklart i Kapittel 4.2. Strengen utformes på JSON-format, og publiseres til et bestemt emne på MQTT-megleren, nærmere bestemt emnet "*anlegg/del_b*". Kodesnutt 12 viser hvordan en meldingspakke ser ut.

```
{
"Batch"   : 1,
"Volume"  : 1,
"Density" : 1,
"Energy"  : 1,
"PV_TT01" : 1,
"PV_TT02" : 1,
"PV_TT03" : 1,
"PV_FT01" : 1,
"PV_FT02" : 1,
"SP_FT01" : 1,
"SP_FT02" : 1,
"UC_P01"  : 1,
"UC_P02"  : 1
}
```

Kodesnutt 12: meldingspakke på JSON-format

Tolkning og logging av meldingspakker

Ved hjelp av Node-RED brukes en MQTT-node som abonnerer på emnet "*anlegg/del_b*". Når JSON-strengen er hentet inn i Node-RED, brukes "Change-nodes" til å formatere dem inn i hvert sitt Node-RED objekt, som videre kan behandles som individuelle målinger. Disse målingene logges så videre opp til InfluxDB ved hjelp av InfluxDB-nodene i Node-RED. I tillegg til å sortere målingene tildeles målingene navn som er lettere å tolke for brukere av anlegget. For eksempel endres "*PV_TT*02" til "Retur".



Figur 44: Node-RED flytkode

```
{
"Retur" : payload.PV_TT02
}
```

Kodesnutt 13: Change-node i Node-RED

Figur 44 viser fire typer noder; MQTT-node (lilla), "Change-nodes" (gul), InfluxDB-node (brun)

og feilsøkingsnode (grønn). “debug 1” er en node som feilsøker de innkommende meldingspakkene.

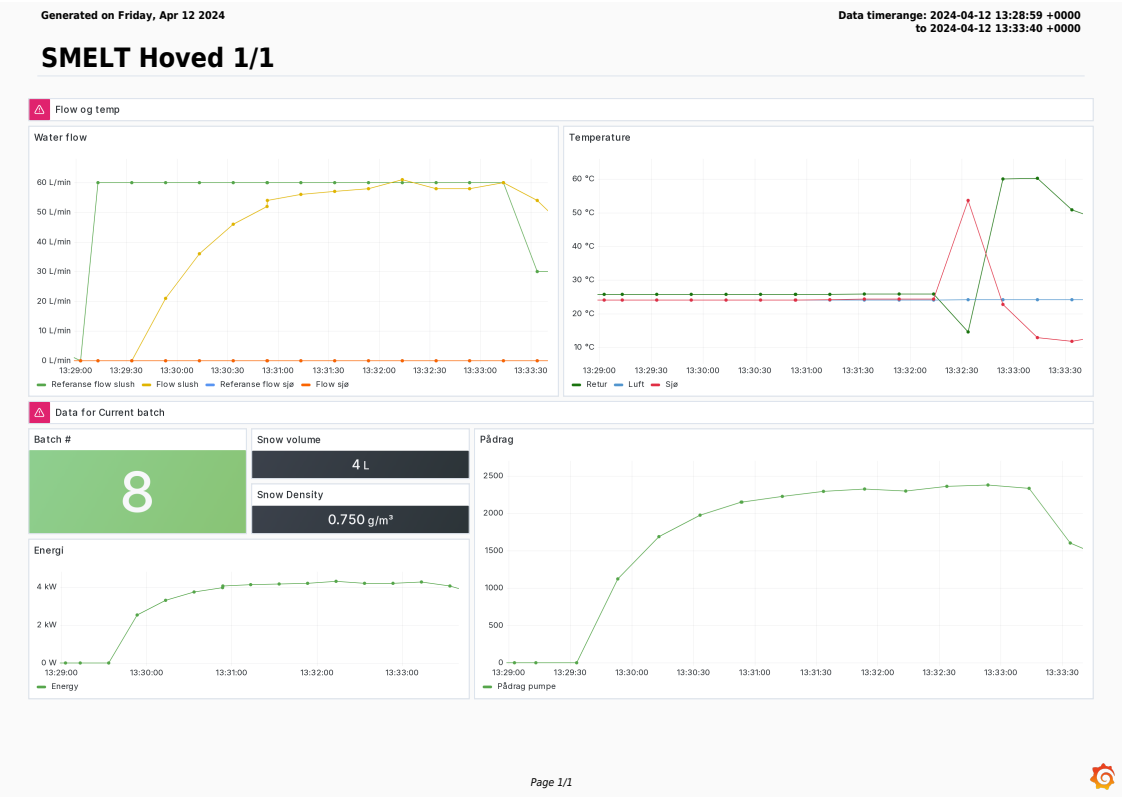
Visualisering av logget data

I Grafana hentes data ut fra InfluxDB ved hjelp av “SQL queries”. Et eksempel på hvordan data hentes ut er illustrert i Kodesnutt 14. Dette gjøres for alle målingene slik at de kan visualiseres i hvert sitt diagram på *dashboardet* i Grafana. For målinger som skal visualiseres sammen, skriver en inn flere målinger i samme SQL-forespørsel. Prosjektgruppen har valgt å representere dataene i fire diagrammer som vist i Figur 45. I første graf visualiseres strømingsdata i faktiske verdier sammenliknet med referanseverdier. Dette gjør det lett å se karakteristikken til sprangresponsen, når en setter referansen. I grafen øverst til høyre er de tre temperaturene for luft, sjø og retur representert. En slik graf kan gi innsikt i temperaturendringer gjennom smelteprosessen. Den mest interessante parameteren her er returtemperaturen siden den gir temperatur på slushen i systemet. Luft og sjøtemperaturen er forventet å ligge forholdsvis stabilt gjennom smelteprosessen.

Dataene i Figur 45, er fra en simulert prosess som skal etterlikne smelteprosessen. Med denne simuleringen har prosjektgruppen testet hvordan dataloggingen vil se ut under virkelige forhold. Grunnen til at det ikke vises til en rapport fra en av testene ute på anlegget er fordi dataloggings-systemet ikke fungerte som forventet under testperioden.

```
SELECT *
FROM "Flow slush"
WHERE
time >= now() - interval '1 hour'
AND
("Flow slush" IS NOT NULL)
```

Kodesnutt 14: SQL query i Grafana



Figur 45: Grafana rapport

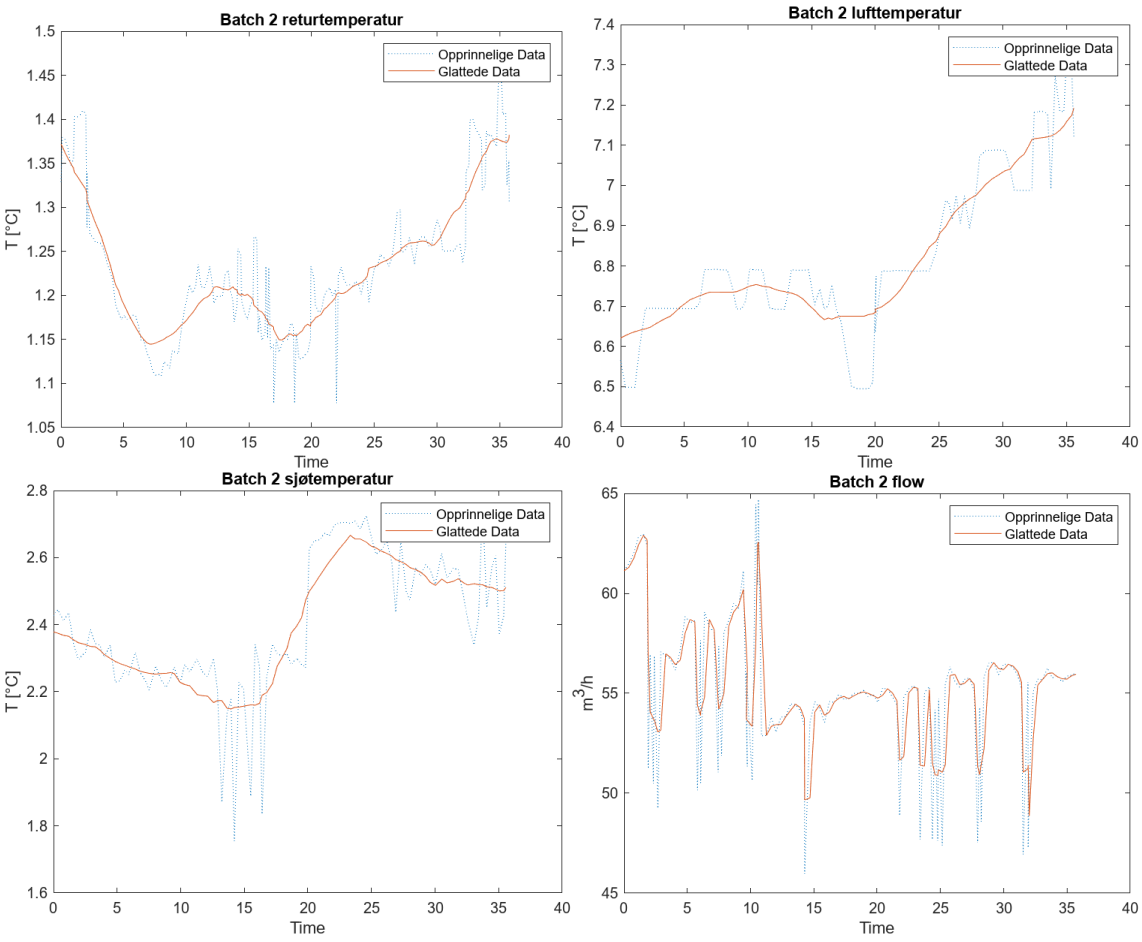
4.6 Målinger

Kapittelet presenterer resultater av datalogging fra testperioden i en oppsummert tabell, samt plottede grafer fra hvert parti.

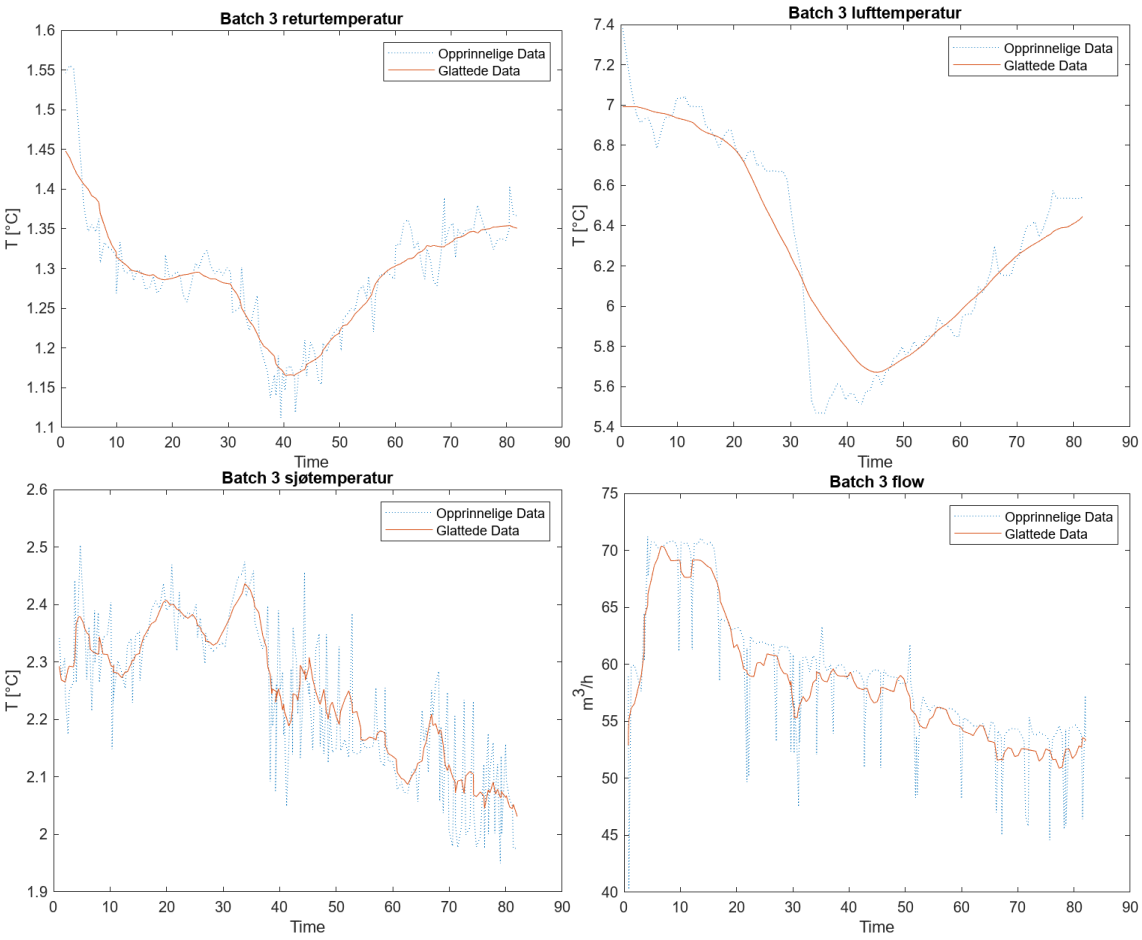
Tabell 11 viser en partivis oppsummering av testresultatene med følgende data: Estimert tidsrom for når snøen er smeltet, loggført snøvolum, median av slush-hastighet og konstant pådrag fra fortøyd båt. I siste kolonne beregnes smeltehastighet som snøvolum delt på tidsrom. Under alle testene var sjøvannstemperaturen og lufttemperaturen tilnærmet lik 5 °C og 6 °C. Dette er basert på loggført værdata i den aktuelle perioden [10], og avviket på dette som vises i Figur 46 og 47 diskuteres nærmere i Kapittel 5.5.

Parti	Dato	Tidsrom [hh:mm – hh:mm]	Snøvolum [m^3]	Slush-hastighet [m^3/h]	Sjøstrøm [RPM]	Smeltehastighet [m^3/h]
1	19.03	11:15 – 13:15	2	25	0	2
2	20.03	10:35 – 10:45	1.5	55	1000	9
3	20.03	11:35 – 11:55	2	58	500	6

Tabell 11: Sammendrag av Testresultater



Figur 46: Målinger fra parti 2



Figur 47: Målinger fra parti 3

I Figur 46 og 47 vises glattede målinger fra parti 2 og 3. Dataen er hentet og behandlet i henhold til metodene i Kapittel 3.4. Utifra grafen for returtemperaturen under parti 2, kan en se at temperaturen synker fra start, før den stiger igjen etter rundt 10 minutter. Dette samsvarer med de fysiske observasjonene av at snø ble dumpet i smeltebingen ved $t = 0$, og at det meste av snøen var smeltet etter ca. 10 minutter.

Utifra grafen for returtemperatur under parti 3, kan en se at temperaturen også synker fra start, før den stiger igjen etter ca. 20 minutter. Disse observasjonene samsvarer også med de fysiske observasjonene, ettersom det meste av snøen var smeltet etter ca. 20 minutter. Man kan også se at temperaturen synker drastisk etter ca. 30 minutter, en tendens som også vises i grafen for lufttemperaturen i det samme tidsrommet. Dette samsvarer med at det begynte å regne i det aktuelle tidsrommet.

4.7 Beregninger

I dette kapittelet presenteres resultater fra beregninger utført på tidsseriedata innsamlet under testperioden. Det er benyttet tre forskjellige fremgangsmåter for å gi bedre innsikt i datasettet, og dermed smelteanlegget i sin helhet. Fremgangsmåtene for de ulike beregningene er beskrevet i detalj i Kapittel 3.4.

MATLAB: Medianberegning og prediksjon

I denne beregningen er det hentet ut medianverdier av målingene fra hvert parti. Figur 48 viser medianverdien for luft- og sjøtemperatur, samt strømning i rør, strømningen i sjøen og smeltehastigheten. I første parti er dataen hentet fra manuell logging, og er derfor gitt i heltall.

AirTemp	SeaTemp	FlowRate	BoatRPM	MeltRate
5	4	25	0	2
6.8211	2.398	55.121	1000	9
6.2967	2.2346	58.114	500	6

Figur 48: Medianverdier fra hvert parti

Dataen fra Figur 48 mates inn i funksjonen "fitlm" fra matlabverktøyet "Statistics and Machine Learning Toolbox". Figur 49 viser estimerte koeffisienter fra regresjonsmodellen.

```
Linear regression model:
    MeltRate ~ 1 + AirTemp + SeaTemp + FlowRate + BoatRPM

Estimated Coefficients:
                   Estimate     SE    tStat    pValue
                   ________    __    _____    ______

    (Intercept)           0    0      NaN      NaN
    AirTemp               0    0      NaN      NaN
    SeaTemp         0.24508    0      Inf      NaN
    FlowRate       0.040787    0      Inf      NaN
    BoatRPM       0.0061641    0      Inf      NaN


Number of observations: 3, Error degrees of freedom: 0
R-squared: 1,  Adjusted R-Squared: -Inf
F-statistic vs. constant model: 0, p-value = NaN
```

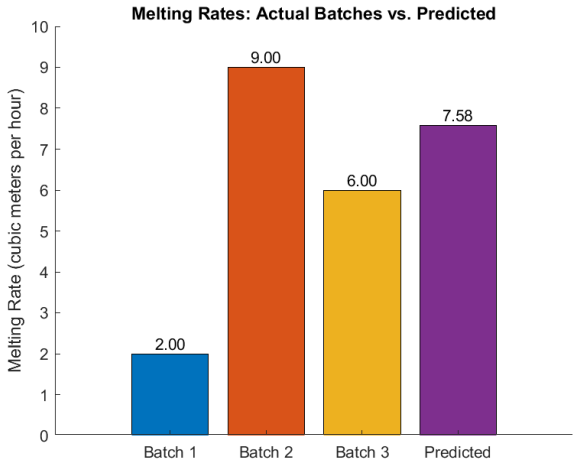
Figur 49: Lineær regresjonsmodell

Med regresjonsmodellen ble det testet å mate inn eksempelverdier for å sjekke om modellen kan forutse smeltehastighet når en justerer omgivelsesvariablene (Sjøtemperatur, lufttemperatur, strømning i rør og strømning i sjø). Figur 50 viser et eksempel på verdier for omgivelsesvariablene en kan bruke i prediksjonen.

```
New data for prediction:

    AirTemp    SeaTemp    FlowRate    BoatRPM
    _______    _______    ________    _______

       6          5          50         700

Predicted melting rate for new conditions: 7.58 cubic meters per hour
```

Figur 50: Ny eksempeldata for prediksjon

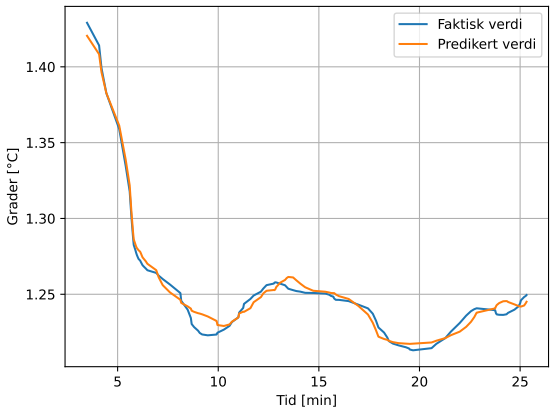
Figur 51 viser den forutsette smeltehastigheten basert på omgivelsesvariablene i Figur 50. I tillegg sammenlikner den også den predikerte smeltehastigheten, med hastighetene fra gjennomførte snøpartier.



Figur 51: Forutsett smeltekapasitet sammenliknet med ekte

Python: Regresjonsanalyse av fullstendig datasett

I denne beregningen er hele datasettet benyttet i en flervariabel lineær regresjonsanalyse ved hjelp av Python maskinlæringsbiblioteket "sklearn". Fremgangsmåten er forklart nærmere i Kapittel 3.4.



Figur 52: Predikert verdi av kombinert returtemperatur

Figur 52 viser et plot som resultat av koden presentert i Kodesnutt 4 i Kapittel 3.4, hvor man ser predikerte verdier for returtemperatur gitt tidsseriedataene samlet inn under testperioden. Den faktiske verdien i plottet er et gjennomsnitt mellom glattet tidsseriedata fra parti 2 og parti 3, og grafen representerer en generell trend hvor det kan tyde på at snøen er smeltet etter rundt 10-25 minutter. Den predikerte verdien i grafen er et resultat av at maskinlæringsmodellen er trent på datasettet, og forsøker å predikere returtemperaturen gitt de samme tilstandene i systemet.

Resultatet av opplæring av modellen er oppsummert i Tabell 12 hvor man ser vektingen av hver enkelt tilstand på systemet, samt R^2-verdi og skjæringspunkt for regresjonslinjen.

Variabel	Koeffisient
luft2	0.0451
sjø2	0.0390
flow2	-0.0085
flow_sjø2	-0.0000
snømengde	0.2320
luft3	0.1764
sjø3	0.0100
flow3	-0.0013
flow_sjø3	0.0000
$\boldsymbol{R^2}$	0.9796
Skjæringspunkt	0.1493

Tabell 12: Koeffisientene til variablene i datasettet

Empirisk modellidentifikasjon - ARIMA

Å kunne simulere dynamikken til hele systemet med en godt tilpasset modell vil kunne gi gode estimater på smeltekapasiteten på anlegget, og dermed være nyttig for optimalisert regulering. Det vil også være nyttig informasjon under kjøring av anlegget hvis modellen kan forutsi smeltekapasiteten en gitt dag, for å ikke overbelaste anlegget. Derfor ble datasettet også matet til en funksjon som automatisk tilpasser data til en ARIMA modell, nærmere beskrevet i Kapittel 3.4. Resultatet er en modell av typen ARIMA(2,0,3), som i praksis er en ARMA modell ettersom parameteren $d = 0$.

Variabel	Koeffisient
luft2	0.0434
sjø2	0.0379
flow2	-0.0083
flow_sjø2	0.0002
snømengde	0.2299
luft3	0.1766
sjø3	-0.0001
flow3	-0.0015
flow_sjø3	0.0001
ar.L1	1.8714
ar.L2	-0.8857
ma.L1	0.4365
ma.L2	0.1480
ma.L3	0.0757

Tabell 13: Oppsummering av ARIMA modell

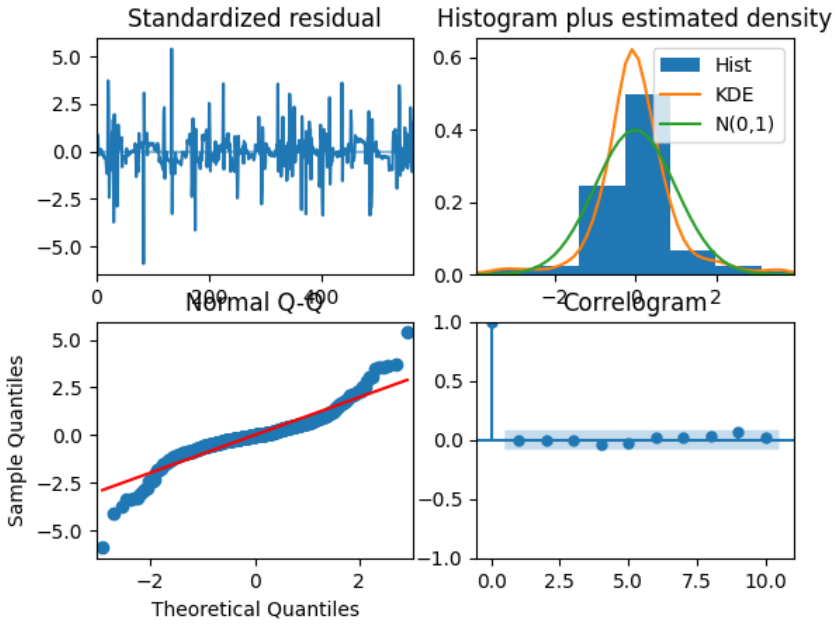
Hvor:

ar.L1	ϕ_1 - Første ledd av autoregressiv del (AR)
ar.L2	ϕ_2 - Andre ledd av autoregressiv del (AR)
ma.L1	θ_1 - Første ledd av glidende snitt del (MA)
ma.L2	θ_2 - Andre ledd av glidende snitt del (MA)
ma.L3	θ_3 - Tredje ledd av glidende snitt del (MA)

Tabell 13 viser en oppsummering av modellen med både koeffisienter for vekting av tilstandene, og for vekting av AR og MA leddene. Ved å sammenligne med koeffisientene i Tabell 12 ser en at det er tilnærmet like verdier for vekting av de ulike tilstandene for begge metodene. Dette er å forvente ettersom det er benyttet samme datasett, og fordi metoden "auto_arima()" blant annet benytter de samme funksjonene fra "sklearn" biblioteket som er brukt i Kapittel 4.7.

$$(1 - 1.8714L + 0.8857L^2)X_t = (1 + 0.4365L + 0.1480L^2 + 0.0757L^3)\epsilon_t \tag{30}$$

I Likning 30 vises ARMA modellen matematisk i henhold til teorien i Kapittel 2.3. For å kvalitetssjekke modellen kan man kjøre funksjonen "plot_diagnostics()" fra "pmdarima" biblioteket for å se hvor godt modellen er tilpasset datasettet. Resultatet vises i Figur 53.



Figur 53: ARIMA Diagnostikk

Figur 53 viser fire ulike grafer som gir et mål på hvor god modellen er sammenlignet med de faktiske dataene. Grafen øverst til venstre viser restene ϵ over tid. Det er ingen tegn til åpenbare sykliske forhold, noe som også bekreftes av grafen nederst til høyre som viser autokorrelasjonen mellom nåverdien $y(t)$ sammenlignet med hver av de foregående verdiene $y(t-k)$. Grafen øverst til høyre viser at restene (KDE) følger en tilnærmet normalfordelt kurve $N \sim (0, 1)$, og dette bekreftes ytterligere av grafen nederst til venstre som viser at restene følger en tilnærmet lineær trend. Oppsummert viser grafene at modellen er godt tilpasset datasettet.

4.8 Sedimentering

Etter smelteprosessen ble smeltevannet pumpet over på en 1000L sedimenteringstank, der smeltevannet ble stående over en natt. Deretter ble det tatt prøver av smeltevannet og sedimentene som hadde lagt seg på bunnen. Det ble også tatt tester av brøytesnø direkte fra deponi. Disse prøvene ble sendt til NGI (Norges Geotekniske Institutt) for analyse.

Hovedfunnene fra prøvene viste at snøen er betydelig forurenset, og at sedimenteringsprosessen er av stor betydning. Prøvene fra restvannet viser at mesteparten av forurensingen vil falle til bunnen. Innholdet av sink og noen PAH-komponenter som blir igjen i restvannet er på et høyere nivå enn foretrukket. Fullstendig rapport fra NGI ligger som vedlegg.

(a) Prøvetaking

(b) Sedimenteringstank

Figur 54: Prøvetaking med sedimenteringstank

5 Diskusjon

I dette kapittelet diskuteres det mulige løsninger på ulike utfordringer prosjektgruppen har vært gjennom, hvorfor den endelige løsningen ble valgt, og eventuelle endringer som kan gjøres etterhvert.

5.1 PLS-programmering

Reguleringsalgoritmer

Under oppstarten av prosjektet ble det diskutert hvorvidt prosjektgruppen skulle programmere en egen PID-regulator med tilhørende ekstrafunksjoner. Det ble konkludert med at dette var unødvendig ettersom det eksisterte gode ferdigløsninger i TIA Portal, "PID_Compact" Likning 17, og fordi oppgaven var omfattende nok på andre punkter. Oppgaven var avhengig av tidlig ferdigstillelse av systemet for å kunne teste anlegget før snøen smeltet og det var derfor viktig å prioritere arbeidsoppgaver riktig i begynnelsen av prosjektet.

Under testperioden var det mange utfordringer som gjorde at reguleringssystemet aldri ble tilstrekkelig testet, og dermed ikke særlig videreutviklet. Dette var fordi det oppstod andre utfordringer som ga et tidspress og det allerede fantes en ferdiglaget Funksjonsblokk for regulering som fungerte bra. Videre var prosjektet under testperioden begrenset til kun én regulerbar motor, nemlig pumpen i slush-bassenget. Dette begrenset også graden av utvikling av reguleringsalgoritmene ettersom det ikke var særlig behov for mer kompliserte oppsett enn en enkel PID-regulator for frekvensomformeren.

Med tanke på brukervennlighet ble det laget en funksjon som forteller når reguleringsavviket på prosessen man regulerer er innenfor ønskede verdier. Denne funksjonen setter variabelen PV_OK høy for den aktuelle regulerte prosessen, som var tiltenkt å benyttes i sekvensdiagrammet for styring av hele prosessreguleringen. For eksempel kunne man benyttet innsamlet data for å sette ønskede verdier for returtemperaturen: $OK\_TIME := \#10m$ og $OK\_E\_\% := 10$, og dermed detektert når snøen er smeltet for større grad av automatisering av anlegget.

Simulert diskret prosess

Tilgang på en simulert prosess har vært nyttig for utvikling på andre områder i prosjektet. For eksempel har det gjort det enklere å verifisere at all kommunikasjon mellom ulike enheter og programvarer fungerer som det skal, samt at visualiseringene både i Grafana og på HMI fungerer. Det har også gjort utviklingen med reguleringsalgoritmene mer håndfast, før hele systemet skulle testes på anlegget under testperioden.

Utvikling av den simulerte prosessen har ført til utfordringer med at PLSen går i "Stopp" som følge av overskriding av maks syklustid. Måten dette ble håndtert på var å plassere funksjonen for simuleringen i en egen organisasjonsblokk, og redusere hvor ofte organisasjonsblokken kjørte.

5.2 IT-løsninger

I oppstartsfasen av prosjektet ble det framstilt noen forslag på hvilke tjenester som kunne bli brukt for kommunikasjon og datalogging fra VisionTech. Disse var Amazon Web Services, MQTT, InFlux DB, Grafana og OPC UA. Etter å ha lest seg opp på tjenestene, ble det undersøkt hvordan en kan logge data fra PLSen opp til en database og deretter visualisere denne dataen. Andre tjenester ble også utforsket, men prosjektgruppen konkluderte med at de listede tjenestene ville bli de mest hensiktsmessige å bruke fordi dette var de mest brukte og mest dokumenterte programmene for formålet til denne oppgaven.

Alternative kommunikasjonsarkitekturer

De aktuelle løsningene prosjektgruppen kom frem til baserte seg på å bruke InfluxDB for tidsserie-database, og Grafana til visualisering. Hvordan data blir logget fra PLS til databasen er forskjellig i de fire løsningene presentert nedenfor:

1. PLS, OPC UA, Telegraph, InfluxDB, Grafana
2. PLS, offentlig MQTT, Telegraph, InfluxDB, Grafana
3. PLS, offentlig MQTT, Node-RED, InfluxDB, Grafana
4. PLS, lokal MQTT, offentlig MQTT, Node-RED, InfluxDB, Grafana

Redgjørelse av OPC UA og Telegraph:

> *OPC UA er nesten blitt en industristandard for industrielle kontrollere, der en vil utveksle data opp til for eksempel en database [70].*
>
> *Telegraph er en integrasjon som kan brukes til å hente, sortere og behandle data som skal inn i databaser, i dette tilfellet InfluxDB. I prosjektgruppens tilfelle ville det oppfylt samme funksjon som Node-RED. Altså å hente data fra enten MQTT eller OPC UA, behandle dataen og plasere den i databasen [71].*

Løsning nr. 1 er å drifte en OPC UA server på PLSen der Telegraph kan hente ut og plassere data i InfluxDB.

Løsning nr. 2 er å bruke LMQTT-klienten på PLSen, der en kan publisere data til en MQTT-megler. Telegraph kan deretter brukes til å abonnere på dataen og plassere den i InfluxDB

Løsning nr. 3 fungerer på samme måte som nr.2 med unntak av at Node-RED blir leddet som abonnerer på data fra MQTT-megleren og plasserer den i Influx DB.

Løsning nr. 4 er som nr. 3, med unntak av at man legger en bro mellom en lokal og offentlig MQTT-megler.

Endelig løsning på kommunikasjonsarkitektur:

Prosjektgruppen bestemte seg i hovedsak for løsning 3, men måtte etterhvert over til løsning 4, da det oppsto problemer med å få koblet PLSen sin LMQTT-klient direkte opp på offentlig nett. Prosjektgruppen har erfaring med Node-RED og MQTT og var derfor hensiktsmessig å benytte dette.

Datalogging og visualisering

Oppdragsgiver anbefalte InfluxDB som database for datalagring, og Grafana for visualisering av data. Både InfluxDB og Grafana er åpen kildekode platformer, noe som gjør det enkelt for bruker å gjøre endringer, utvikle og oppskalere systemet for videre bruk etter endt bacheloroppgave.

Kommunikasjonen mellom InfluxDB og PLSen gjennom MQTT skapte utfordringer. Under testperioden sviktet tilgangen på AWS, og dermed tilgangen til Node-RED. Sensordata som ble sendt fra PLSen kom ikke frem til Node-RED og videre til InfluxDB og Grafana. Det ble laget en ny Node-RED server på anlegget, gjennom den skybaserte løsningen FlowFuse. Ettersom den gamle Node-RED koden ikke var tilgjengelig fysisk på anlegget, måtte det lages en ny. Det ble en tidkrevende prosess å sette opp ny server og skrive koden på nytt. MQTT-tilkoblingen gjennom FlowFuse var ikke stabil, og fungerte ikke optimalt på anlegget, og ble dermed ikke nyttig. Løsningen for datalogging ble å ta skjermbilder direkte fra TIA Portal, som nevnt i Kapittel 3.3.

5.3 HMI

Oppdragsgiver hadde noen spesifikasjoner for hvordan HMI skulle se ut og fungere. Disse inkluderte visning av prosessverdier med fargekoding for aktuelle verdier, muligheten til å styre programmet enkelt fra startsiden, og at alle sensorer skulle vise prosessverdier. Utifra dette og kommunikasjon internt i prosjektgruppen ble det utviklet et brukergrensesnitt som skulle brukes i testperioden og utvikles videre ved fremtidig drift av anlegget.

I testperioden ble det tydelig hva som fungerte og hva som ikke fungerte i brukergrensesnittet, som at det er enklest å styre og navigere i systemet fra operatørpanelet. Det er også mulig å styre systemet i drift fra en tilkoblet PC ved å monitorere i TIA Portal. Dette vil derimot være mer tungvint og fungere tregere ettersom det krever mer prosessorkraft fra PLSen. Det er vanskelig å navigere gjennom hele TIA-programmet for å feilsøke. Ved noen tilfeller har dette vært nødvendig, men ved å tilrettelegge for spesifikk feilsøking på HMI-panelet kan dette gjøres enklere og raskere.

På HMI-panelet finner man mye informasjon og justerbare verdier for å enkelt kunne navigere systemet. Dette er fordi operatørpanelet fortsatt skal være ryddig og oversiktlig. På startsiden ligger all informasjon og monitorering som brukeren trenger for å styre systemet. Det er også lagt til en kontrollside der det ligger ytteligere informasjon for feilsøking. Alarmvinduet er også et viktig hjelpemiddel for å feilsøke i programmet. Dette forklares ytteligere under Kapittel 5.6 om alarmsystemet.

5.4 Empiriske modellberegninger

Oppgaven fra oppdragsgiver var blant annet å forske på optimal hastighet på slush- versus båtpropellhastighet, og på denne måten legge grunnlaget for optimalisert regulering for smeltekapasitet og energiforbruk. Det ble derfor gjort flere beregninger for å få gode svar utifra dataene fra testperioden.

Mangel på gode nok data er et gjennomgående problem for de ulike metodene for beregning av smeltekapasiteten. Ytelsesmetrikkene fra for eksempel ARIMA modellen forteller oss kun om modellen er godt tilpasset dataene, ikke om det er en god modell på virkeligheten.

Noen av koeffisientene fra modellereingene gjort i Python med fullstendig datasett motsier deler av beregningene på medianverider i MATLAB, samt observasjoner fra virkeligheten. For eksempel ble det observert at strømning i sjøen hadde stor innvirkning på smeltekapasiteten, mens Python beregningene viser det motsatte. Dette kan forklares med en svakhet i metoden datasettet ble benyttet på, nemlig at data fra de ulike partiene ble matet til regresjonsmodellen som separate variabler. For å modellere systemet bedre burde hver variabel fra partiene samles i samme tidsserie, for eksempel at variabelene “sjø2” og “sjø3” kombineres.

Ved videreføring av prosjektet kan det likevel være hensiktsmessig å ta i bruk de samme metodene for beregning av smeltekapasitet dersom man får tilgang på mer og bedre data. Det skal være enkelt å tilpasse beregningene dersom man får flere sensorer som måler tilstander av interesse, som igjen kan ytterligere forbedre resultatene. For eksempel kunne en sensor for strømningsmåling i sjøen vært gunstig for bedre prosessmodellering.

5.5 Instrumentering

VisionTech og prosjektgruppen bygde skap på VisionTech sine lokaler utifra instrumenteringsliste og informasjon fra oppdragsgiver i RebelGarden. Det ble da laget et styre- og hovedstrømskjema. Alle kabler er merket. Skapet som ble kjøpt inn er større enn det som egentlig er nødvendig for oppsettet for dette prosjektet, fordi da er det mulig å skalere opp uten å kjøpe nytt skap.

Dimensjonering til pumpe

Under testingen ble automasjonstavlen installert på anlegget. Alle sensorer, med unntak av pH- og konduktivitetssensorene, og pumpen ble koblet opp der. Dette skapte utfordringer siden motorvernet, frekvensomformeren og kabling var dimensjonert for en effekt på 5,5 kW. Etter at skapet var montert på anlegget, oppdaget prosjektgruppen at pumpen var på 11 kW i stedet for de forventede 5,5 kW. Det ble fort funnet ut at ved fullt pådrag, så slo motorvernet ut ettersom det ikke var dimensjonert høyt nok. I tillegg hadde motoren en uvanlig høy startstrøm på 152,9 A sammenlignet med den nominelle strømmen på 22 A, noe som tilsvarer en startstrøm på 7 ganger den nominelle. For å unngå at motorvernet slo ut på grunn av den brå startstrømmen, kunne man ha gradvis økt startstrømmen ved å ha en lenger opprampningstid i frekvensomformeren. For å øke stabiliteten i systemet ble motorvernet derfor byttet ut med et nytt motorvern som håndterte høyere strømmer, fra 10 A til 25 A.

Koblingsboks mellom pumpe og frekvensomformer

Pumpen på anlegget hadde en koblingsboks mellom pumpe og skapet, for å kunne styre pumpen manuelt. I denne lå det motorvern og en sikring. Motorvernet skapte utfordringer fordi det mest sannsynlig har vært vannintrengning i koblingsboksen og den er blitt kortsluttet. Dette gjorde at systemet ikke klarte å kjøre pumpen. Det krevde mye feilsøking for å finne ut av hvor denne feilen lå. En HMI-side for feilsøking av parameterne til frekvensomformeren ble blant annet laget. Koblingsboksen måtte til slutt kobles helt ut av systemet.

Strømningsmåler

Strømningsmåleren viste urealistiske verdier på grunn av en feil. Sensorverdiene som strømningsmåleren sendte til PLSen samsvarte ikke med det strøminingsmålerdisplay viste. Årsaken til dette var mangelen på felles jord mellom strømningsmåleren og PLSen. Strømningsmåleren fikk strømforsyning fra 230 V AC, og PLSen fra 24 V DC. De hadde dermed ingen felles referansepunkt. Fra datablad til strømningsmåleren vet man at den har mulighet til å ta imot 24 V DC, men det må da loddes direkte på kretskortet. Vanlighvis ville dette blitt gjort på verkstedet, men siden problemet ble oppdaget på anlegget, måtte det brukes en alternativ løsning. Løsning som ble valgt å bruke var å dra en ledning fra jordingsterminal på strømningsmåler til minuspolen på 24 V forsyningen. Da fikk PLSen og strømningsmåleren et felles referansepunkt.

Temperatursensorer

Input modulen "AI 4xRTD/TC HF" har innganger nok til fire RTD-sensorer, hvorav tre av disse er i bruk. De er av typen "RTD PT100". Det var usikkerhet rundt om de skulle kobles opp med fire, tre eller to kabler, siden databladene viste flere oppkoblingsalternativer. I konfigurasjon i TIA Portal skal det skrives inn hvordan sensorene er kablet, hva slags type metall (i dette tilfellet er det PT100) og hvor lang leder som er satt opp til til sensorene for å kompensere for ekstra motstand. Under avlesning av målingene når alle sensorene var plassert i samme temperatur, ga de forskjellige verdier. Det var ikke noen ekstern sensor tilgjengelig for å kryssjekke målingene. Det var dermed usikkert om det bare var noen av sensorene som var konfigurert riktig eller om alle var feil. Da ble det utforsket mye rundt hvordan en skulle få like verdier på temperatursensorene. På grunn av tidspress ble dette problemet ikke løst under testperioden. En mulig forklaring kan være

at det ikke var riktig kompensert for kabellengdene fra skap til sensor. De ulike temperatursensorene viste i romtemperatur verdiene 23°C, 23,6°C og 26°C. Dette er uheldig for de gjennomførte målingene, siden avviket vil forplante seg i videre beregninger. Til tross for avviket ser det ut til at temperaturvariasjonene stemmer siden motstanden i RTD-sensorer er lineære.

5.6 Alarmsystem

Den endelige løsningen for alarmstruktur fungerte godt, og bruken av DB, FB, og FC i kombinasjon med HMI-alarms gjør at løsningen tar hensyn til skalerbarhet. Ved å organisere all kode som omhandler avvik og alarmer i en FC, blir programmet oversiktlig.

Sensorene er analoge og måtte i utgangspunktet ha to analoge alarmer pr. sensor som varsler om prosessverdien er over eller under forventet verdi. For å minimere antall alarmer lagde prosjektgruppen en funksjonsblokk for å detektere feil, "Sensor_Alarm [FB7]". Dette gjør at antall sensoralarmer halveres, noe som er en fordel dersom det skulle være nødvendig å implementere flere sensorer ved oppskalering av systemet.

Fargene i alarmlisten er valgt etter vanlig standard, oransje og rød for aktive varslinger (warnings) og alarmer (error). Tydelige farger er viktig for å ikke skape misforståelser ved potensielt kritiske situasjoner, og er en del av prinsippet om høy ytelse brukergrensesnitt.

5.7 Testperiode

Prosjektgruppen måtte gjøre mange endringer i instrumentering, PLS-program og HMI under testperioden. Alt dette førte til forsinkelser i arbeidet som skulle gjøres på anlegget i testperioden. Anlegget ble satt opp midlertidig for testingen for å utforske smeltekapasiteten for det spesifikke anlegget. Det var kostbart å drive anlegget med innleid utstyr og arbeidskraft. Det var derfor viktig å løse utfordringene som oppsto underveis så raskt som mulig. Mange parter skulle samarbeide; kranbil, snøtransport, prosjektleder og prosjektingeniører. Dette førte til at hvis det oppsto problemer på ett arbeidsområde, måtte ofte resten vente til problemet ble løst. På anlegget var det bare mulig å gjøre endringer i PLS-programmet på én tilkoblet PC. Dette var lite tidseffektivt.

Sedimenteringsprosessen

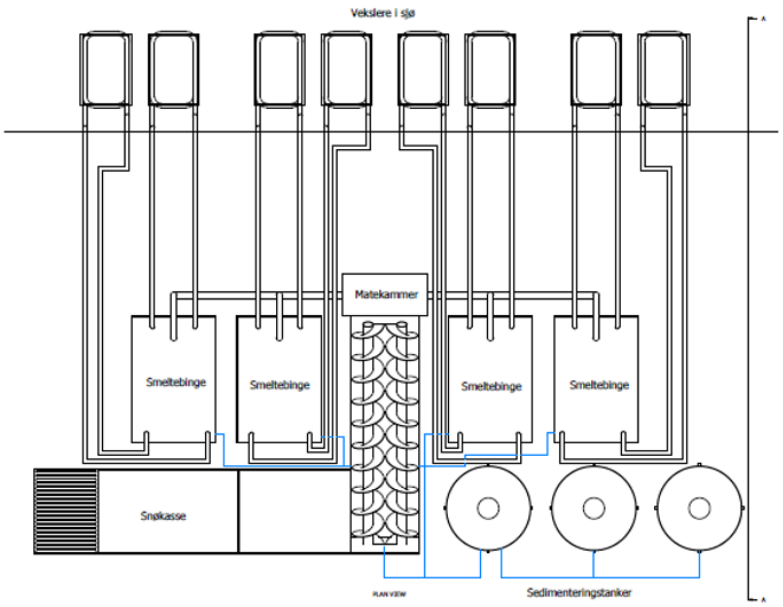
Ifølge tester fra NGI, er løsningen med en sedimenteringstank av stor miljømessig betydning. Prosessen gjør at restvannet og bunnslammet kan gjenvinnes hver for seg, og hindrer at forurensing havner i naturen. Restvannet kan gjenvinnes som "bruksvann", eller tappes i sjøen. Slammet kan sendes til nærliggende miljøpark for videre håndtering. Brøytesnø inneholder ofte mye strøgrus, og ved et fullskalert anlegg vil muligheten for å gjenbruke denne være aktuell. I følge målingene, bør restvannet derimot gjennom enda en renseprosess før det brukes videre eller havner i sjøen. Dette innebærer tilsetting av flokkuleringsmiddel, og vil forhåpentligvis fjerne de siste restene av sink og PAH-komponenter. Ved oppskalering av anlegget bør det gjøres flere tester for å bekrefte eller avkrefte om denne prosessen er nødvendig. Dersom nye målinger også viser rester av sink i restvannet, vil det fortsatt være nødvendig med tilsetting av flokkuleringsmidler.

5.8 Forbedringer/Videreutvikling

Bacheloroppgaven har vært del av et større prosjekt, noe som har medført at arbeidsoppgavene endret seg underveis grunnet tid og kapasitet. Når prosjektgruppen er ferdig med sitt arbeid, vil oppgaven videreføres til VisionTech. Det blir da deres oppgave å videreutvikle programmet for oppskalering. Prosjektgruppen foreslår at man i nær fremtid gjør følgende forbedringer: bedre kalibrering av temperatursensorene, automatisere sedimentering (fase 3, Figur 23), gjøre mer matematiske beregninger på smeltekapasiteten, samt videreutvikle visualisering i HMI og Grafana.

Prosjektet SMELT er delt opp i flere deler, der bachelorgruppen har vært mest involvert i del B i fase 2 som vises i Figur 23. Det skal være mulig å slå sammen det arbeidet som ble gjort i denne bacheloroppgaven med de resterende delene. Muligheten for oppskalering har også vært et viktig kriterium gjennom hele prosjektperioden. På bakgrunn av dette er det forsøkt å bruke standardisert programvare, forklart nærmere i Kapittel 3.2.1. For at oppskalering skal være mulig er det blitt tilrettelagt for ulike oppjusteringer, blant annet ved at skapet som ble kjøpt inn er dimensjonert større enn det som var nødvendig for denne spesifikke bacheloroppgaven. Det er plass til å sette inn flere komponenter, som for eksempel en ekstra frekvensomformer.

I neste fase av SMELT-prosjektet skal det være fokus på oppskalering. Målet er å utvide anlegget slik at det kan håndtere all brøytesnø i større byer, med spesielt fokus på Oslo og Trondheim. Kjernen i prosjektet er fortsatt å utnytte energien fra temperaturendifferansen mellom snø og sjø. Videre utvikling av anlegget avhenger av ønsket smeltekapasitet. Det skal implementeres en snø- og iskvern for å effektivisere smelteprosessen. Det er også satt opp en plan med ulike tiltak som kan justers. Dette innebærer blant annet installasjon av flere varmevekslere eller justeringer i volumet i veksleren.



Figur 55: Skisse av mulig anlegg ved oppskalering

5.9 FNs bærekraftsmål

Hovedformålet med SMELT-prosjektet er "klimavennlig håndtering av brøytesnø" [1]. Da er det viktig å ta hensyn til FNs bærekraftsmål. I dette delkapittelet skal de ulike aspektene av løsningen vurderes opp mot de aktuelle bærekraftsmålene beskrevet i Kapittel 1.4.

Snøsmelteanlegget vil minimere, eller i bestefall erstatte helt, bruken av snødeponi langt utenfor byene. Dette vil minimere CO_2-utslipp i forbindelse med transport med store og tunge kjøretøy, og dermed bidra til å nå bærekraftsmål 11 og 13. I tillegg vil det hindre at forurensing og miljøgifter fra brøytesnøen havner i naturen, og forstyrrer økosystemet og artsmangfoldet i nærområdet. Dersom deponier er plassert i nærheten av drikkevannskilder, vil dette også påvirke drikkevannkvaliteten. I følge testresultatene fra NGI, er brøytesnøen betydelig forurenset, og bekrefter dermed at det har reelle konsekvenser å lagre snø i deponi. Ved å hindre at brøytesnø blir liggende lenge i naturen, vil dette ha positiv innvirkning på bærekraftsmål 6 og 15.

Ved å smelte snøen vil den også frigjøre plass i byområdet. Da vil det ikke lenger være nødvendig å dumpe snøen rett i sjøen for å unngå snøkaos i byene, slik det har vært flere eksempler på i 2024 [4]. Dette vil gagne livet i havet, som unngår forurensing, giftige kjemikalier og annet avfall som havner i sjøen sammen med brøytesnøen. Dermed vil konseptet med et smelteanlegg bidra til å forbedre bærekraftsmål 14.

Selve driften av anlegget må også ta hensyn til bærekraftsmålene. Ved å utnytte temperaturforskjellen mellom sjøvann og snø for smelting, bruker anlegget minimalt med energi. Daglig drift vil dermed ikke ha store miljøutslipp. Ved oppskalering vurderes det å ta i bruk fjernvarme for å gjøre prosessen raskere. Med dette alternativet vil det fortsatt redusere utslippene som kommer av transporten av snøen betydelig.

Sedimenteringsprosessen er viktig for at smeltevannet trygt kan slippes ut i naturen eller gjenvinnes. Sedimenteringsprosessen tar hensyn til bærekraftsmål 6, både delmål 6.3 ved å rense vannet før det tappes i sjøen, og 6.4 dersom det kan gjenvinnes som bruksvann. Delmål 14 blir også ivaretatt, da enten rent vann eller ikke noe vann blir tappet i havet. Livet i havet unngår dermed miljøgiftene, forurensingen og avfallet som kommer med dumping av snø direkte i havet.

Figur 56: FNs bærekraftsmål [12]

6 Konklusjon

Dette kapittelet tar for seg hvordan problemstillingen er besvart ved å sammenfatte alle resultatene fra prosjektet. I tillegg settes oppgaven i et perspektiv, der gruppen svarer på hvordan veien videre for prosjektet kan se ut.

Oppdragsgiveren får et produkt som inneholder de ønskede kravene etterspurt i oppgaveteksten. Automasjonstavle med frekvensomformer og PLS ble bygget og brukt i testperioden. Skapet er såpass stort at det er mulig å legge til flere moduler i ettertid uten store endringer. Det er laget PLS-program med reguleringsalgoritmer for regulering av pumpen, og et fungerende alarmsystem. HMI-panelet på skapet gjør det enkelt å styre og overvåke prosessen ved anlegget. Oppsettet til den ferdige programvaren for PLS og HMI, har en slik struktur at det enkelt skal kunne videreutvikles ved oppskalering av prosjektet. Data fra systemet logges via sky i InfluxDB. Prosessverdier kan overvåkes i sanntid via det nettleserbaserte programmet for visualisering, Grafana. Det gjør at man kan overvåke prosessen uten å være tilstede på anlegget.

Svarene fra testperioden viste at smelteprosessen var effektiv, og det ble funnet gode parametere for energieffektiv smelting. En viktig observasjon fra testen var at sirkulasjonen i sjøvannet hadde stor påvirkning på smeltehastigheten. Testresultatene av sedimentet og restvannet etter smelting ga bekreftelse på at sedimenteringsprosessen har hensiktsmessig effekt. Totalt sett viste testperioden at konseptet fungerte slik som ønsket, og at dette vil være en mulig løsning for håndtering av brøytesnø som samtidig ivaretar flere av FNs bærekraftsmål. Ved stor nok oppskalering kan et slikt anlegg erstatte bruk av snødeponi og dumping i sjø.

Oppdragsgiver fra Rebel Garden AS var fornøyd med utføringen av testperioden og beregningene av smeltekapasitet gjort i etterkant. Rebel Garden og VisionTech vil bruke produktet og beregningene prosjektgruppen har levert videre i de neste fasene av SMELT-prosjektet.

Referanser

[1] Trondheim Kommune. *SMELT- Snøbehandling utført Miljøvennlig med Energinøytraliserende Lagring og Teknologi*. 2023. URL: https://www.trondheim.kommune.no/tema/veg-vann-og-avlop/veg/vedlikehold-og-drift/smelt--prosjekt-for-smart-handtering-av-broytesno/ (sjekket 29. jan. 2024).

[2] Svendsen Elin. *Kommuner dumper fortsatt snø i sjøen*. 2022. URL: https://www.kommunal-rapport.no/miljoe/kommuner-dumper-fortsatt-sno-i-sjoen/148098!/.

[3] Miljødirektoratet. *Håndtere snøbrøyting*. Miljødirektoratet. 2020. URL: https://www.miljodirektoratet.no/ansvarsomrader/forurensning/Haandtere-sno-broyting/ (sjekket 13. feb. 2024).

[4] NRK Vestfold og Telemark. «Larvik dumper snø i sjøen». I: (jan. 2024). URL: https://www.nrk.no/vestfoldogtelemark/larvik-dumper-sno-i-sjoen-1.16700170.

[5] RBnett. *Får ikke dumpe snø i sjøen lenger: Nå hoper det seg opp*. 2024. URL: https://www.rbnett.no/nyheter/i/O84mEl/faar-ikke-dumpe-snoe-i-sjoeen-lenger-naa-hoper-det-seg-opp (sjekket 24. apr. 2024).

[6] Hagen Arne og Langeland Arnfinn. «Polluted snow in southern Norway and the effect of the meltwater on freshwater and aquatic organisms». I: *Environmental Pollution (1970)* 5.1 (1973), s. 45–57. ISSN: 0013-9327. DOI: https://doi.org/10.1016/0013-9327(73)90055-4. URL: https://www.sciencedirect.com/science/article/pii/0013932773900554.

[7] Valmot Odd Richard. *Snøsmelteeksperten Terje gjør at Oslo går rundt når hovedstaden drukner i snø*. Teknisk Ukeblad. 2018. URL: https://www.tu.no/artikler/snosmeltelekteren-terje-gjor-at-oslo-gar-rundt-nar-hovedstaden-drukner-i-sno/415861 (sjekket 13. feb. 2024).

[8] Høilund Anders. *Kommunens snøsmelter «S/s Terje» skal rense snøen for forurensning, men spyr selv ut avgasser og eksos*. Vårt Oslo. 2024. URL: https://www.vartoslo.no/bydel-gamle-oslo-bymiljoetaten-hanne-sofie-fremstad/kommunens-snosmelter-s/s-terje-skal-rense-snoen-for-forurensning-men-spyr-selv-ut-avgasser-og-eksos/530311 (sjekket 13. feb. 2024).

[9] Gjelsvik Øystein Hansgård. *Vedtak om utslippstillatelse etter forurensningsloven for snøsmelteanlegget SS Terje - NCC Constructions AS*. Statsforvalteren. 2023. URL: https://www.statsforvalteren.no/siteassets/fm-oslo-og-viken/horinger-og-kunngjoringer/snosmelteanlleget-ss-terje/vedtak-om-utslippstillatelse-etter-forurensningsloven-for-snosmelteanlegget-ss-terje---ncc-constructions-as.pdf (sjekket 13. feb. 2024).

[10] sea temperature. *Sjøvannstemperatur i Trondheim*. 2024. URL: https://no.seatemperature.net/current/norway/trondheim-sor-trondelag-norway.

[11] Holte Runar. *SMELT – Klimavennlig håndtering av snø fra bygatene*. Oppgaveforslag. Trondheim: VisionTech AS, 2024.

[12] FN-sambandet. *FNs bærekraftsmål*. 2024. URL: https://fn.no/om-fn/fns-baerekraftsmaal (sjekket 3. mai 2024).

[13] Lienhard John H. IV og Lienhard John H. V. *A Heat Transfer Textbook*. 5. utg. Phlogiston Press, 2020. URL: https://ahtt.mit.edu/wp-content/uploads/2020/08/AHTTv510.pdf.

[14] Vijzelaar John H. og Lynglund Magnar. *PLS: Hva er det, og hvordan er det oppbygd?* Jan. 2020. URL: https://ndla.no/nb/subject:1:8c5a9fdd-4fa4-456b-9afe-34e7e776b4e7/topic:ce841519-de73-4349-870f-2240e5276bc0/resource:3eca6f64-0408-46bc-aa3c-827c7ead3352 (sjekket 9. mai 2024).

[15] Muthukrishnan Vidya. *Programmable Logic Controllers (PLCs): Basics, Types & Applications*. Electrical4U. 2024. URL: https://www.electrical4u.com/programmable-logic-controllers/ (sjekket 19. mai 2024).

[16] Danfoss. *Hva er en frekvensomformer?* 2024. URL: https://www.danfoss.com/nb-no/about-danfoss/our-businesses/drives/what-is-a-variable-frequency-drive/ (sjekket 9. mai 2024).

[17] Mohan Ned, Robbins William P. og Undeland Tore M. *Power electronics*. John Wiley & Sons Inc, 2002.

[18] Kothari D. P. og Nagrath I. J. *Electric Machines*. McGraw Hill, 2018.

[19] Unknown author. *Electric Motor Components*. Unknown year. URL: https://power-mi.com/sites/default/files/elearning/vibration_analysis/10/en%20electric%20motor%20components.jpg (sjekket 2. mai 2024).

[20] Coman Nicolae. *Induction Motor*. https://commons.wikimedia.org/wiki/File:Induction_motor.png. 2015.

[21] Ramprasath E. og Manojkumar Purvi. «Modelling and Analysis of Induction Motor using LabVIEW». I: *International Journal of Power Electronics and Drive Systems* 5 (feb. 2015), s. 344–354. DOI: 10.11591/ijpeds.v5i3.7220.

[22] Rita, Refvik. *Temperaturmåling: Slik virker PT100-elementet*. Mar. 2013. URL: https://www.tu.no/artikler/temperaturmaling-slik-virker-pt100-elementet/218877 (sjekket 3. mai 2024).

[23] GMS Instruments. *What is a PT100 Resistance Thermometer?* Jan. 2021. URL: https://gms-instruments.com/no/what-is/what-is-a-pt100-resistance-thermometer/ (sjekket 3. mai 2024).

[24] IQS Directory. «Magnetic Flow Meters». I: (). URL: https://www.iqsdirectory.com/articles/flow-meter/magnetic-flow-meter.html (sjekket 3. mai 2024).

[25] Bjørvik Kåre og Hveem Per. *Reguleringsteknikk*. Eget forlag, 2014.

[26] *Reguleringsteknikk*. 2022. URL: https://snl.no/reguleringsteknikk (sjekket 24. apr. 2024).

[27] Seborg Dale E., Edgar Thomas F., Mellichamp Duncan A. og Doyle Francis J. III. *Process Dynamics and Control*. fourth. John Wiley & Sons, 2016.

[28] Siemens. *SIMATIC STEP 7 Basic/Professional V19 and SIMATIC WinCC V19*. 2024. URL: https://cache.industry.siemens.com/dl/files/862/109826862/att_1163651/v1/STEP7_WinCC_Engineering_V19_enUS.pdf (sjekket 11. apr. 2023).

[29] *Linear Regression in Matrix Form*. Purdue Department of Statistics. URL: https://www.stat.purdue.edu/~lingsong/teaching/2018spring/topic3.pdf (sjekket 19. mai 2024).

[30] *Coefficient of determination*. Wikipedia. URL: https://en.wikipedia.org/wiki/Coefficient_of_determination (sjekket 20. mai 2024).

[31] *Autoregressive integrated moving average*. 2024. URL: https://en.wikipedia.org/wiki/Autoregressive_integrated_moving_average (sjekket 30. apr. 2024).

[32] *Lag operator*. Wikipedia. URL: https://en.wikipedia.org/wiki/Lag_operator (sjekket 2. mai 2024).

[33] Iordanova Tzveta. «An Introduction to Non-Stationary Processes». I: *Investopedia* (2022). URL: https://www.investopedia.com/articles/trading/07/stationary.asp#citation-1 (sjekket 30. apr. 2024).

[34] Jain Garima og Mallick Bhawna. «A Study of Time Series Models ARIMA and ETS». I: *SSRN Electronic Journal* (jan. 2017). DOI: 10.2139/ssrn.2898968. URL: https://ssrn.com/abstract=2898968.

[35] ABB. *Industrial IT System 800xA 5.1 - Engineering*. 2015. URL: https://library.e.abb.com/public/81478a314e1386d1c1257b1a005b0fc0/2101127.pdf (sjekket 24. apr. 2024).

[36] *IEC 61131-3*. 2024. URL: https://el3.no/automasjon/pls/ps-grunnkurs/iec-61131-3/ (sjekket 24. apr. 2024).

[37] Siemens. *Using blocks to structure your program*. 2014. URL: https://support.industry.siemens.com/cs/mdm/107623221?c=70596025099&lc=en-AO (sjekket 26. apr. 2024).

[38] Dietrich Shawn. *Understanding PLC Tags: Controller Scope vs. Program Scope*. 2023. URL: https://control.com/technical-articles/understanding-plc-tags-controller-scope-vs-program-scope/ (sjekket 11. mai 2024).

[39] Instrumentation Tools. *OB1 Main Cyclic Organization Block*. 2017. URL: https://instrumentationtools.com/main-cyclic-organization-block/ (sjekket 26. apr. 2024).

[40] Siemens. *Function (FC)*. 2015. URL: https://support.industry.siemens.com/cs/mdm/107623221?c=29182113291&lc=en-AO (sjekket 26. apr. 2024).

[41] Siemens. *Function block (FB)*. 2014. URL: https://support.industry.siemens.com/cs/mdm/107623221?c=71643534091&lc=en-AO (sjekket 26. apr. 2024).

[42] Siemens. *Data block (DB)*. 2015. URL: https://support.industry.siemens.com/cs/mdm/107623221?c=71643549835&lc=en-AO (sjekket 26. apr. 2024).

[43] Siemens. *Which organization blocks can you use in STEP 7 (TIA Portal)?* 2013. URL: https://support.industry.siemens.com/cs/document/40654862/which-organization-blocks-can-you-use-in-step-7-(tia-portal)-?dti=0&lc=en-DE.

[44] Øverby Harald. *brukergrensesnitt*. 2023. URL: https://snl.no/brukergrensesnitt (sjekket 9. mai 2024).

[45] Hollifield Bill mfl. *The High Performance HMI Handbook*. Houston, TX: Plant Automation Services, 2008. ISBN: 978-0977896912.

[46] Hollifield Bill. *High Performance Graphics to Maximize Operator Effectiveness*. PAS Global, 2017.

[47] Advansia. *IT- og OT-sikkerhet*. 2024. URL: https://www.advansia.no/tjenester/it-og-ot-sikkerhet (sjekket 9. mai 2024).

[48] Red Hat. *What is operational technology (OT)?* Aug. 2022. URL: https://www.redhat.com/en/topics/edge-computing/what-is-ot (sjekket 9. mai 2024).

[49] Cisco. *How do OT and IT differ?* URL: https://www.cisco.com/c/en/us/solutions/internet-of-things/what-is-ot-vs-it.html (sjekket 9. mai 2024).

[50] *Ethernet*. Techtarget. URL: https://www.techtarget.com/searchnetworking/definition/Ethernet (sjekket 14. mai 2024).

[51] *Ethernet frame*. Wikipedia. URL: https://en.wikipedia.org/wiki/Ethernet_frame (sjekket 14. mai 2024).

[52] *Ethernet Cables Explained*. Eaton. URL: https://tripplite.eaton.com/products/ethernet-cable-types (sjekket 14. mai 2024).

[53] *WHAT IS PROFINET? – PROFINET EXPLAINED*. 2021. URL: https://us.profinet.com/profinet-explained/ (sjekket 24. apr. 2024).

[54] *PROFINET EXPLAINED*. PROFIBUS & PROFINET International (PI). URL: https://www.profinet.com/profinet-explained/technology-description (sjekket 11. mai 2024).

[55] PROFIBUS Nutzerorganisation. *Filnavn: $PROFINET_System_Description_engl_2018_Update.pdf$*. 2018. URL: https://de.profibus.com/downloads/profinet-technology-and-application-system-description.

[56] Siemens AG. *SINAMICS G120, G120P, G120C, G120D, G110M Fieldbuses: Function Manual*. 04/2018. Siemens AG, Division Digital Factory. 2018. URL: https://support.industry.siemens.com/cs/ww/en/view/109485727.

[57] Siemens Industry Online Support. *SINAMICS G: Controlling a Speed Axis with the "SINA_SPEED" Block*. Versjon V1.0. 2017. URL: https://support.industry.siemens.com/cs/ww/en/view/109485727.

[58] *MQTT: The Standard for IoT Messaging*. https://mqtt.org/. MQTT.org, 2022. (Sjekket 16. apr. 2024).

[59] *Information technology - Message Queuing Telemetry Transport (MQTT) v3.1.1*. Standard. Geneva, CH: International Organization for Standardization og International Electrotechnical Commission, 2016.

[60] InfluxData. *Why Time Series*. Technical Report. InfluxData, 2023. URL: https://get.influxdata.com/rs/972-GDU-533/images/why%20time%20series.pdf (sjekket 20. apr. 2024).

[61] w3schools. *SQL Syntax*. 2024. URL: https://www.w3schools.com/sql/sql_syntax.asp (sjekket 3. mai 2024).

[62] Siemens GA. *Solutions for Powertrain, System Manual HMI Lite*. 2021. URL: https://cache.industry.siemens.com/dl/files/058/109795058/att_1058864/v1/SystemManual_HMILite_V16_EN.pdf (sjekket 3. mai 2024).

[63] WebPlotDigitizer. *Opensource computer vision assisted software to help extract numerical data from images of plots, maps and much more*. URL: https://automeris.io/WebPlotDigitizer.html (sjekket 29. apr. 2024).

[64] automeris.io. URL: https://automeris.io/WebPlotDigitizer.html.

[65] *Statistics and Machine Learning Toolbox Documentation*. MathWorks. URL: %7Bhttps://se.mathworks.com/help/stats/index.html?s_tid=CRUX_lftnav%7D (sjekket 11. mai 2024).

[66] scikit-learn. *scikit-learn Linear Models*. URL: https://scikit-learn.org/stable/modules/linear_model.html (sjekket 29. apr. 2024).

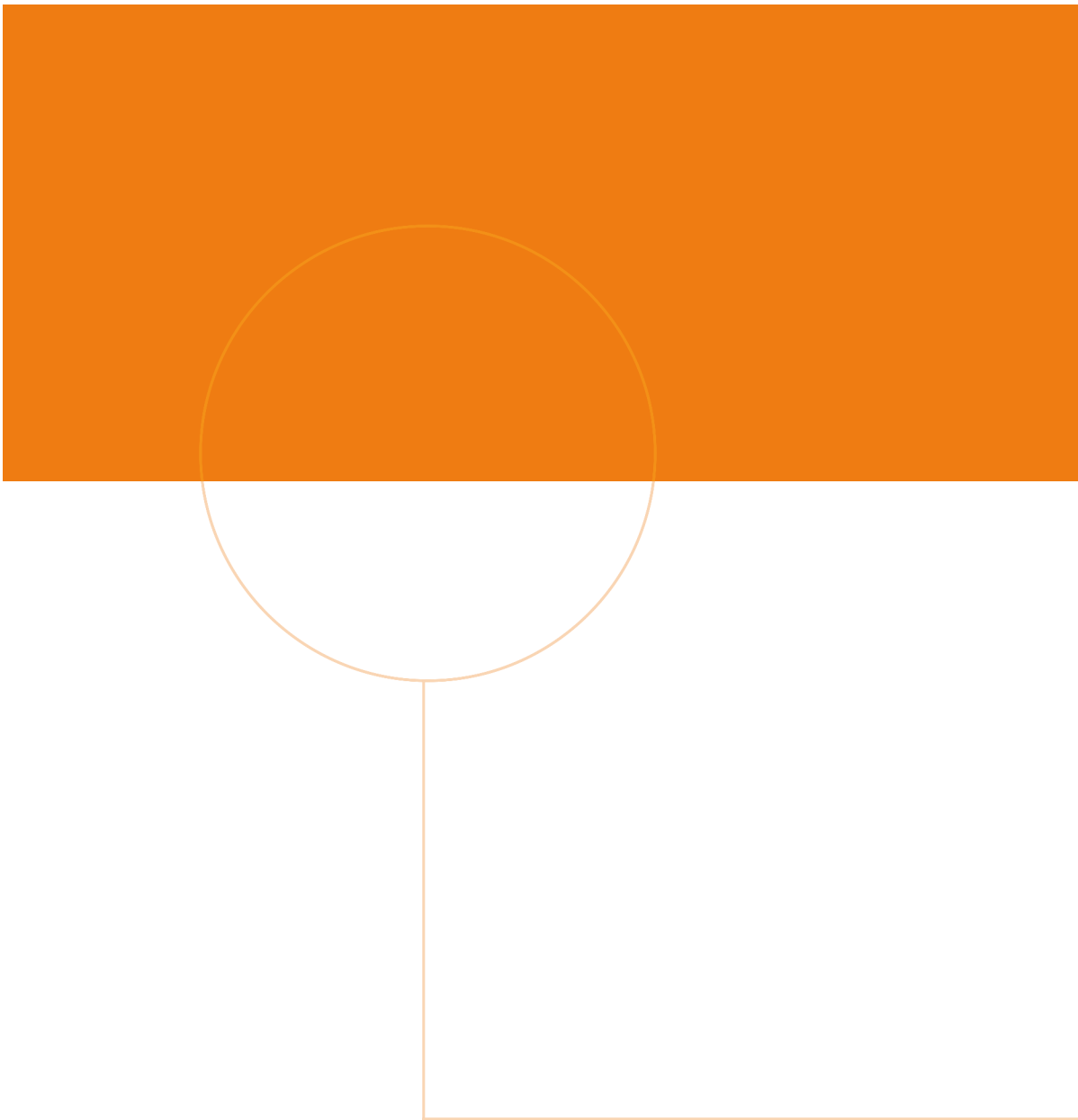
[67] scikit-learn. *scikit-learn Machine Learning in Python*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html#sklearn.model_selection.train_test_split (sjekket 29. apr. 2024).

[68] Alfredsen Roger, Erlandsen Dag Finn og Corneliussen Steinar. *Norm for symboler i driftskontrollsystemer for VA-sektoren*. Rapport. Norsk Vann BA, 2007.

[69] Wikipedia. *Grotesk (skrift)*. Nov. 2022. URL: https://no.wikipedia.org/wiki/Grotesk_(skrift) (sjekket 9. mai 2024).

[70] OPC Foundation. *Unified Architecture*. 2019. URL: https://opcfoundation.org/about/opc-technologies/opc-ua/ (sjekket 11. mai 2024).

[71] InfluxData. *Telegraf: Time Series Data Collection*. 2024. URL: https://www.influxdata.com/time-series-platform/telegraf/ (sjekket 11. mai 2024).



NTNU
Kunnskap for en bedre verden

VisionTech